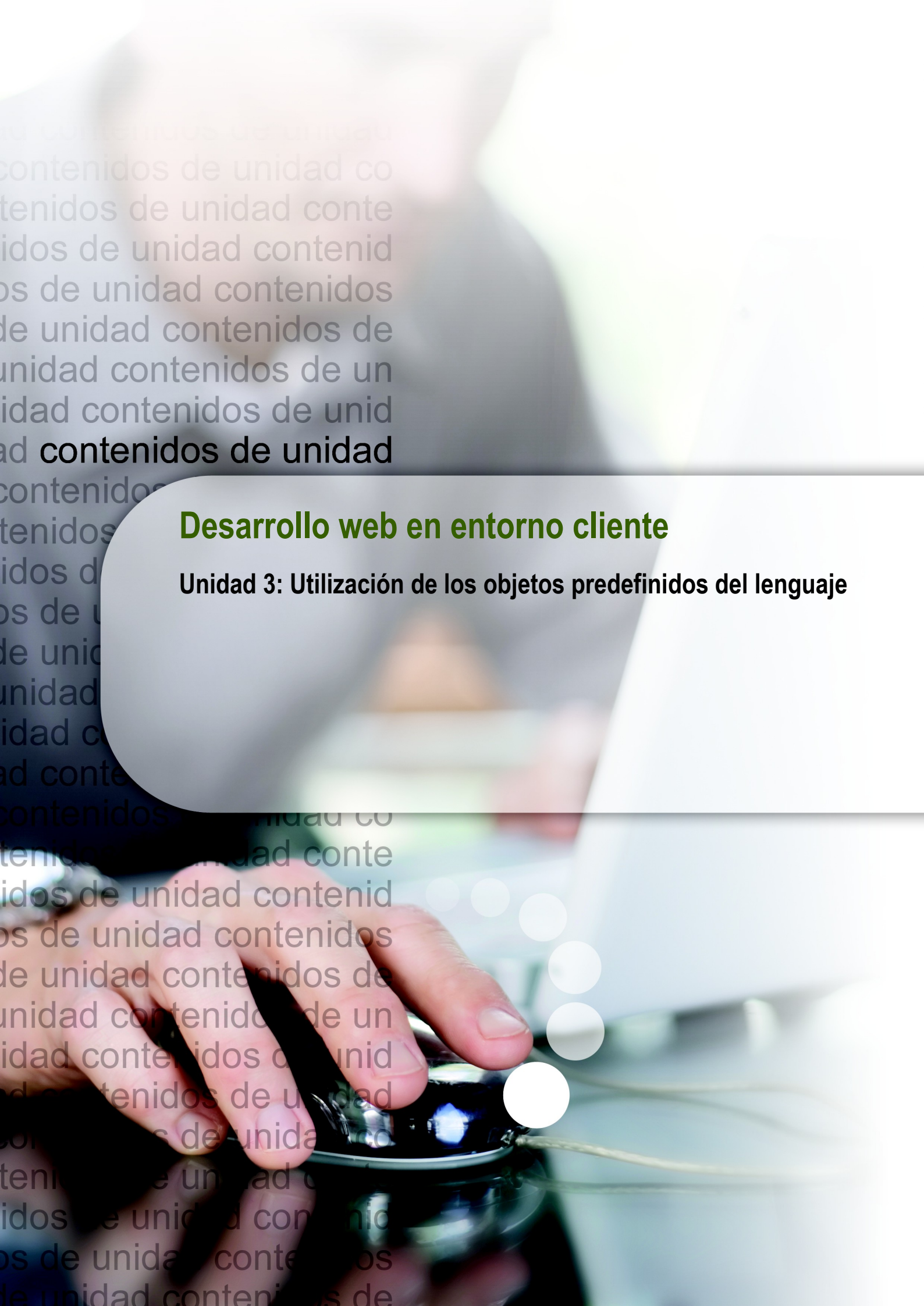


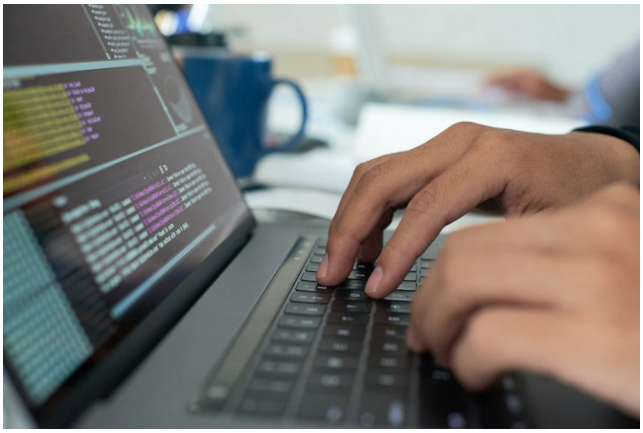
Desarrollo web en entorno cliente

Unidad 3: Utilización de los objetos predefinidos del lenguaje

ÍNDICE

INTRODUCCIÓN

En el capítulo anterior, estudiamos en profundidad el lenguaje **JavaScript**, su sintaxis y funcionamiento. Durante este capítulo, nos centraremos en las posibles interacciones que podemos realizar a través del lenguaje.



Estas **interacciones** las realizaremos a base de objetos, los cuales empezaremos a descubrir.

Entre estos objetos veremos las diferencias entre los **objetos nativos del lenguaje** y los que no lo son. Haciendo especial inciso en aquellos que nos permiten la interacción con e. navegador, y por tanto, con el **usuario final**.

Veremos, además, como trabajar con distintos elementos del navegador, como son los **marcos** y las **ventanas**, e incluso como poder mostrar código **HTML desde JavaScript**.

Para esta unidad y posteriores es importante tener ciertos conocimientos del **lenguaje HTML**, que se trabaja en otros módulos del ciclo.

OBJETIVOS / CAPACIDADES

En esta unidad de aprendizaje, las capacidades que más se van a trabajar son:

- ✔ **Utilizar lenguajes, objetos y herramientas, interpretando las especificaciones para desarrollar aplicaciones Web con acceso a bases de datos.**

PROYECTO DE LA UNIDAD

Antes de empezar a trabajar el contenido, te presentamos la **actividad** que está relacionada con esta unidad de aprendizaje. Se trata de un **caso práctico** basado en una **situación real** con la que te puedes encontrar en tu puesto de trabajo. Con esta actividad se evaluará la puesta en práctica de los **criterios de evaluación** vinculados al resultado de aprendizaje que se trabaja en esta unidad. Para realizarla deberás hacer lo siguiente: lee el enunciado que te presentamos a continuación, dirígete al área general del módulo profesional, concretamente a la actividad de evaluación que se encuentra dentro de esta unidad, allí encontrarás todos los detalles sobre fecha y forma de entrega, objetivos... A lo largo de la unidad irás adquiriendo los conocimientos necesarios para ir elaborando este proyecto.

Enunciado:

Empezando la página de encuestas

En la empresa están encantados con nuestros pequeños mini-proyectos acerca de la imagen corporativa de la empresa. Así que nos han encargado una tarea un poco más grande al respecto, que en un futuro servirá para hacer encuestas a los empleados, acerca de la usabilidad de la página.

Se trata de modificar el archivo HTML adjunto, para que, mediante código JavaScript, haga lo siguiente:

- Muestre la hora actual, en formato dd/mm/aaaa. Puedes usar el objeto Date.
- Muestre las medidas actuales de la ventana.
- Al clicar en el enlace, pregunte al usuario mediante la instrucción prompt cómo se llama.
- Al responder a la pregunta, ha de abrir una nueva ventana saludando al usuario, con una medida de 300x300.

- Almacene el nombre en una cookie, para la próxima visita, y lo muestre por defecto en el prompt.

El programa tiene que estar comentado y probado en al menos 2 navegadores.

```
<!DOCTYPE HTML>
<html>

<body>
    <p id="fecha">- Aqui irá la fecha - </p>
    <p id="medidas">- Aqui irán las medidas - </p>

    <a href="javascript:preguntarNombre()">Introduce el
nombre</a>

</body>

</html>
```

1. UTILIZACIÓN DE OBJETOS. OBJETOS NATIVOS DEL LENGUAJE

A estas alturas ya te habrás dado cuenta de que, en JavaScript, la mayor parte del tiempo estamos trabando con **objetos**, o al menos, la forma en que JavaScript trata los elementos o los tipos de datos con los que trabaja, es como si fueran objetos.

A continuación, veremos algunos aspectos relevantes sobre la **utilización de objetos**:

- **Método**: hemos visto que si queremos escribir alguna cosa por la consola, podemos utilizar el método **log**() del objeto consola. Si queremos, por el contrario, escribir cualquier cosa en el documento, lo que haremos será utilizar el método **write**() del objeto **documento**. Ambos son **objetos** de JavaScript.

*Muchos de los objetos propios de **JavaScript**, es decir, aquellos que nacen con la propia **sintaxis** del lenguaje, ya los conocemos, como por ejemplo el objeto Array, Date, Math…*

- **Tipos de objetos**: en concreto, en JavaScript disponemos de los siguientes tipos de objetos nativos:
  - ✔ Array.
  - ✔ Boolean.
  - ✔ Date.
  - ✔ Error.
  - ✔ Global.
  - ✔ JSON.
  - ✔ Math.
  - ✔ Number.
  - ✔ Operators.
  - ✔ RegExp.

- ✔ Statements.
- ✔ String.

En los siguientes capítulos veremos más usos concretos de estos objetos de JavaScript, como, por ejemplo, la interacción con el navegador y el usuario, la generación de código HTML desde JavaScript o la gestión de las ventanas.

2. INTERACCIÓN CON EL NAVEGADOR. OBJETOS PREDEFINIDOS ASOCIADOS. ELEMENTOS PARA INTERACCIÓN CON EL USUARIO

A parte de los objetos nativos, JavaScript tiene una utilidad de vital importancia, que es la de manipular la estructura de documentos (DOM) HTML que son interpretados por un navegador o user-agent (BOM).

Los conceptos y el trabajo en profundidad del DOM y el BOM los trabajaremos en profundidad más adelante.

Lo importante ahora, es saber que JavaScript se relaciona con estos elementos, y lo manipula todo como si de **objetos** se tratase.

- Así que los **elementos del DOM y del BOM** también serán objetos de JavaScript:
  - ✔ Attributes.
  - ✔ Console.
  - ✔ Document.
  - ✔ Elements.
  - ✔ Events.
  - ✔ Event Objects.
  - ✔ Geolocation.
  - ✔ History.
  - ✔ HTMLCollection.
  - ✔ Location.
  - ✔ Navigator.
  - ✔ Screen.
  - ✔ Style.
  - ✔ Window.

- ✔ Storage.

➔ **Objeto window**: **window** es, quizás, uno de los objetos más **importante** de todos, o al menos, de él cuelgan otros elementos.

Este objeto es la representación del **propio navegador**. Por tanto, todo lo que nos ofrece el navegador, como, por ejemplo, las propiedades (alto, ancho, estructura del documento…) o los métodos, lo gestionaremos con él.

Normalmente, cuando utilizamos algún método de un objeto usamos la notación por punto (objeto.método()), pero en el caso de window, al ser la raíz de todo, no es necesario.

```
document.write(innerWidth); // Ancho del area. Equivale a clientWidth.
document.write(innerHeight); // Alto del area. Equivale a clientHeight.
document.write(outerWidth); // Ancho de la ventana.
document.write(outerHeight); // Alto de la ventana.
document.write(screenX); // Posición X del navegador respecto a la pantalla.
document.write(screenY); // Posición Y del navegador respecto a la pantalla.
```

El objeto window tiene otras propiedades. Alguna de ellas son objetos por sí mismos como la propiedad **document, console, history, location, navigator o screen**.

Vídeo: objeto window

Visualiza este vídeo en el que se habla sobre la propiedad location del objeto window.
https://www.youtube.com/embed/0gSY_w1QVLo

3. GENERACIÓN DE TEXTO Y ELEMENTOS HTML DESDE CÓDIGO. MANIPULACIÓN DE ELEMENTOS HTML DINÁMICAMENTE

Ya sabemos que JavaScript es un lenguaje de programación web **dinámico** que nos permite que nuestros programas se ejecuten en páginas web no estáticas.

También hemos visto que, para ejecutar código de script, usualmente insertamos este código en páginas **HTML**, para que pueda ser ejecutado. Es decir, la página HTML de alguna manera actúa de **anfitriona** de nuestro código dinámico.

Esto carecería de mucho sentido si no tuviéramos mecanismos para que nuestro código pudiera interactuar de alguna manera con el código HTML, que es precisamente lo que veremos ahora.

Imaginemos que tenemos una página HTML con un **botón**, pulsando el cuál, haremos una pregunta a usuario y obtendremos el valor de la respuesta, para finalmente, mostrarlo en nuestra página HTML.

Esto es lo que haremos en el ejemplo siguiente, mediante la instrucción **innerHTML** de JavaScript:

Código: ejemplo innerHTML

```
<html>
<body>
        <button  onclick="  preguntarNombre()">Pulsa  para  introducir  tu
nombre</button>
    <p id="nombre"></p>
```

```
    <script>
        function preguntarNombre() {
            var person = prompt("¿Cómo te llamas?", "Fulanito");
            if (person != null) {
                document.getElementById("nombre").innerHTML = "¡Hola " +
person + "! ¿Qué tal todo?";
            }
        }
    </script>
</body>
</html>
```

➔ **Función del botón**: como vemos, tenemos una página HTML sencilla, con un botón y un párrafo. El botón llamará a una función de JavaScript, que mediante la instrucción prompt, preguntará el nombre al usuario.

La gracia de todo esto está en la línea:

```
document.getElementById("nombre").innerHTML = "¡Hola " + person
+ "! ¿Qué tal todo?";
```

*Lo que se hace en esta línea es coger el identificar del párrafo HTML, en este caso "nombre", y mediante la instrucción "**innerHTML**", cambiar su contenido, que pasa de estar en blanco a contener el nombre escrito por el usuario.*

➔ **document.write()**: de esta manera vemos que desde el código JavaScript estamos accediendo a las **propiedades** creadas en HTML, para ver o modificar su contenido.

Otra forma más básica de creación de código HTML mediante JavaScript es con la instrucción **document.write()**, que ya hemos visto anteriormente.

Vídeo: innerHTML

Visualiza este vídeo en el que hablamos sobre la instrucción innerHTML.

Actividad de aprendizaje 1: primera tarea en la empresa

Una vez superada la fase de prácticas de la empresa, hemos sido contratado oficialmente. Nuestra primera tarea será trabajar en la página web corporativa de la empresa. Desde la empresa están haciendo un estudio de cuáles son los colores que representan mejor la marca, por eso nos piden:

- Crear una página web en la cual el usuario puede escoger el color de fondo de la página.
- Se hará mediante unos enlaces que habrá en la esquina superior derecha.
- Necesitamos un mínimo de 4 colores, y siempre tiene que haber la opción de volver al color original (blanco).

Entrega el archivo .html en el espacio habilitado para ello.

Cuestionario

Lee el enunciado e indica la opción correcta:

¿Qué método se usa para escribir por consola?

a. write().

b. log().

c. innerHTML().

Lee el enunciado e indica la opción correcta:

¿Cuál de los siguientes no es un tipo de objeto nativo?

a. Array.

b. Date.

c. Window.

Lee el enunciado e indica la opción correcta:

¿Con qué instrucción podemos saber qué elemento de HTML tiene cierto ID?

a. getElementById.

b. getElement.

c. getID.

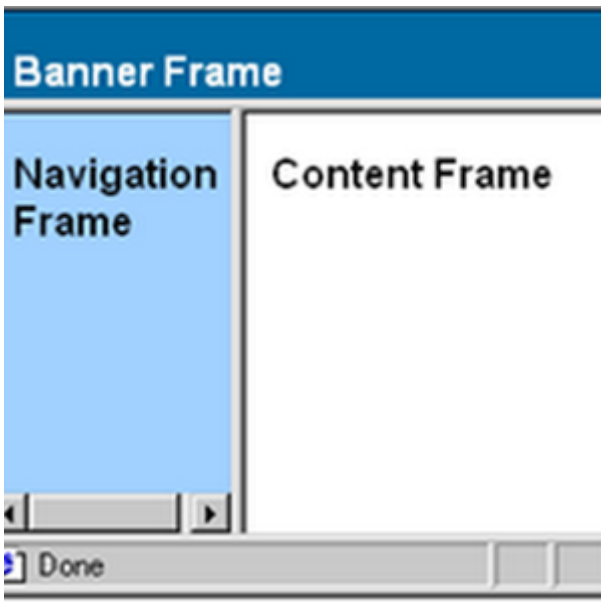
4. GESTIÓN Y CREACIÓN DE MARCOS

Antes de adentrarnos en todo lo relacionado con los **marcos en JavaScript**, haremos un breve repaso a que se refieren.

DESTACADO

Un marco, o frame, en HTML no es más que una estructura que nos permite dividir la ventana de nuestro navegador en diferentes áreas, cada uno con un tamaño específico, que puede ser fijo, o dinámico en función del tamaño del navegador.

Cada marco es **independiente** del resto, pudiendo incluso contener sus **propias barras de desplazamiento** si así lo queremos.



A continuación, veremos algunos aspectos relevantes sobre los **marcos**:

- **Crear marco**: para la creación de una página web con marcos, debemos sustituir la etiqueta HTML body, por la etiqueta frameset, pudiendo especificar, por ejemplo.
  - Si se trata de **divisiones verticales u horizontales**, o incluso una mezcla de ellos.

- ✔ Las **medidas de los distintos marcos de la página**, que pueden ser fijas, o dinámicas según el porcentaje del espacio disponible, por ejemplo.

➔ **Contenido**: cada marco tiene un contenido diferente, que vendrá indicado por la **etiqueta frame**, especificando la página HTML que lo contiene.

```
<html>
<head>
    <title>Marcos</title>
</head>
<frameset rows="200,*">
    <frame src="pagina1.html" />
    <frame src="pagina2.html" />
</frameset>
</html>
```

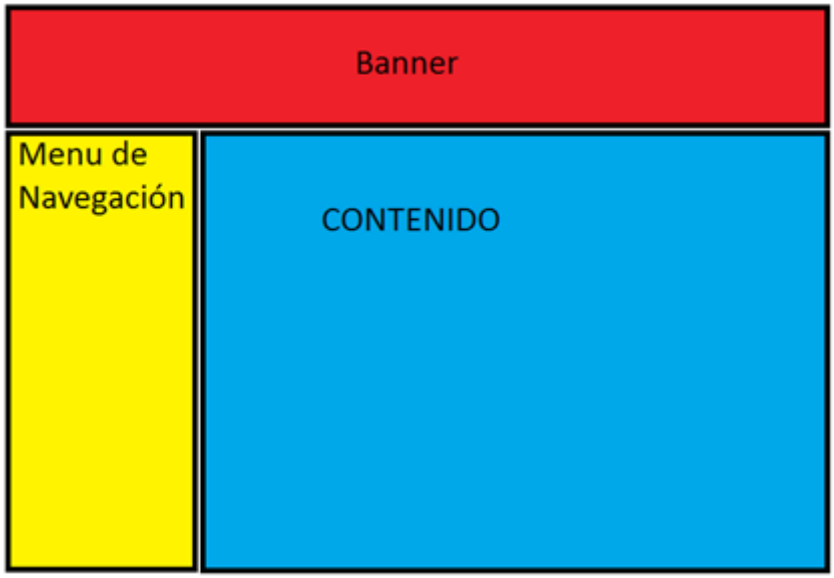
En el ejemplo, dividimos una página de manera **horizontal**, con dos marcos que hacen referencia a dos páginas HTML externas. El marco de arriba tendrá 200 puntos de **altura**, y el de abajo ocupará todo el espacio que quede **disponible** en función de la altura total de la ventana (indicado con el asterisco).

5. MARCOS ANIDADOS

Ahora que ya sabemos lo que son los marcos, vamos a ver una peculiaridad de estos, que los hace especialmente potentes a la hora de gestionar la información en una página web, es lo que se denominan, **marcos anidados**.

Como en muchos ámbitos de la programación, usamos el término anidado, para referirnos a algo que está dentro de una estructura similar de nivel superior. Por ejemplo, los bucles anidados, que significa que hay un bucle dentro de otro.

- **Marcos anidados**: con los marcos anidados sucede exactamente lo mismo. Tener varios marcos anidados, significa que hay **marcos dentro de otros marcos**, como por ejemplo en la imagen siguiente.



Aquí, vemos que hay 3 marcos, aunque realmente unos están dentro de otros. En concreto hay 2 marcos principales, divididos **horizontalmente**. El primero es el rojo, y el otro está compuesto de otro **frameset**, que a su vez contiene 2 marcos, esta vez **verticales**, que son el amarillo y el azul.

- **Código**: el código podría ser algo parecido a esto:

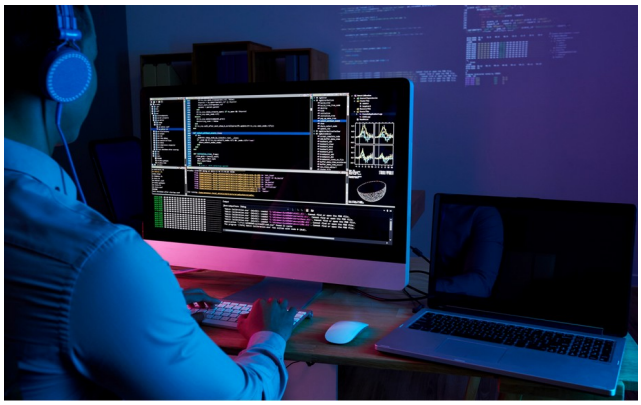
```
<html>
<head>
    <title>Marcos anidados</title>
</head>
<frameset rows="15% ,*">
    <frame src="Banner.html" name="banner" />
    <frameset cols="25%, *">
        <frame src="Menu.html" name="menu" />
        <frame src="Contenido.html" name="contenido" />
    </frameset>
</frameset>
</html>
```

En este caso el marco del banner ocupará el 15% de la altura de la página. En el caso de los dos marcos anidados, se repartirán el espacio con un 25% y 75% en modo vertical.

Vemos como para anidar marcos, basta con poner etiquetas frameset dentro de otras del mismo tipo. Hay que tener especial cuidado para que la estructura nos quede de la manera que realmente queremos.

6. EJECUCIÓN DE CÓDIGO ENTRE MARCOS. COMUNICACIÓN ENTRE MARCOS

Cuando una página tiene varios **marcos**, es importante que se puedan **comunicar** de alguna manera entre ellos. JavaScript nos da las herramientas para poder ejecutar código entre marcos. Esto significa que, desde cualquiera de los marcos de una página, podemos acceder a otros marcos, y ejecutar código allí.



Para todo lo referente al trabajo con marcos en JavaScript usaremos el **objeto window**, que ya hemos visto anteriormente. En este objeto hay algunos métodos y propiedades interesantes:

- La principal es que desde una ventana podemos acceder a la ventana padre, es decir, la ventana que contiene esta ventana. Para ellos usaremos la **propiedad parent**.

```
window.parent
```

DESTACADO

Esta propiedad es la que nos permite que, desde un marco, podamos acceder a otros. Bastará con acceder al padre, y desde ahí a los marcos de este. Para ello usaremos la propiedad frames[], que nos dará una array con los marcos que tiene una ventana.

Podemos acceder a un marco mediante su índice o mediante su nombre. Así podemos hacer:

```
window.parent.frames[0]
window.parent.frames["menu"]
```

- También podemos usar la **propiedad top**, para acceder a la ventana raíz de nuestra página:

`window.top`

A partir de aquí podemos acceder a las distintas propiedades del frame, como, por ejemplo:

- ✔ **document**: el documento del frame.
- ✔ **frame**: frames anidados.
- ✔ **frames[]**: array con todos los frames anidados.
- ✔ **length**: cantidad de frames dentro del frame.
- ✔ **name**: nombre del frame.

7. APLICACIONES PRÁCTICAS DE LOS MARCOS

A día de hoy, con las nuevas tecnologías, especialmente **HTML5**, se considera a los marcos como algo **obsoleto**. Sin embargo, aún existen multitud de páginas que los utilizan.

Las principales **aplicaciones** de los marcos son:

- **Estructuración de la página**: permite que nuestra página principal quede estructurada de manera fácil en otras páginas, más sencillas.
- **Maquetación de la página**: a través de las medidas de los marcos, bien sean fijas, por porcentajes u ocupando el tamaño restante, podemos maquetar de forma bastante sencilla una página con distintos subapartados.
- **Independencia de páginas**: podemos hacer que las distintas páginas que conforman los marcos sean independientes entre ellas, facilitando así su diseño.
- **Comunicación entre páginas**: además, y como hemos visto, si en algún momento lo necesitamos podemos hacer que esas páginas se comuniquen entre sí mediante JavaScript.

A pesar de todas estas aplicaciones, como hemos dicho, se está convirtiendo en una tecnología obsoleta, debido sobre todo a algunas de sus **desventajas**:

- Dificultad de los **motores de búsqueda** para indexar páginas que contienen marcos.
- Ocupan un **espacio fijo** en la pantalla, que no puede ser ocupado por otros elementos.
- Los botones de **avanzar o retroceder** en el navegador no funcionan adecuadamente.
- La **usabilidad y accesibilidad** de las webs con marcos no es la que debería ser hoy en día.

Es importante, aún así, conocer la estructura de marcos y la interacción con JavaScript porque es probable que en nuestra vida de programadores sigamos encontrándonos con bastantes páginas que aún hacen uso de ellos.

Cuestionario

Lee el enunciado e indica la opción correcta:

¿Qué etiqueta HTML no debemos usar cuando trabajamos con marcos?

a. head.

b. html.

c. body.

Lee el enunciado e indica la opción correcta:

¿Qué palabra usamos para definir cada uno de los marcos?

a. frameset.

b. frame.

c. marc.

Lee el enunciado e indica la opción correcta:

¿Si queremos dividir en marcos horizontalmente usaremos?

a. cols.

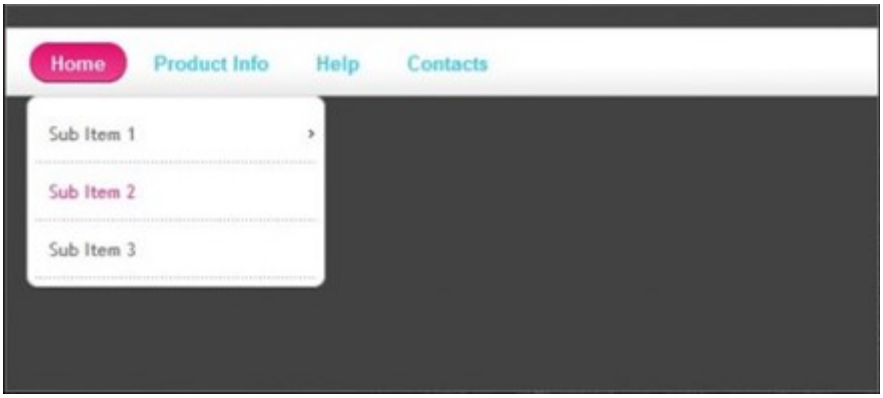
b. rows.

c. Es indiferente.

8. GESTIÓN DE MENÚS

Una de las formas más habituales de navegar por páginas web, es mediante el uso de menús, tanto fijos como desplegables.

A través de una combinación de código HTML, CSS y sobre todo JavaScript podemos crear estos menús de una forma más o menos sencilla.



Hay varias **opciones**, entre las que destacan:

- Que la página HTML y el CSS lleven el peso de los menús, y **JavaScript** sólo se encargue de mostrarlos, ocultarlos o **desplegarlos**.
- Que sea **JavaScript** el que se encargue de toda la creación de los menús.

➔ **Opción 1**: vamos a ver la primera opción. La idea es la siguiente:

- ✔ 1. Crearemos nuestra página web HTML, con los menús desplegados.
- ✔ 2. Mediante estilos de CSS, ocultaremos los menús que no sean de primer nivel, dejando sólo a la vista los de primer nivel.
- ✔ 3. Crearemos el código JavaScript, que se ejecutará cuando pulsemos sobre un menú (o pongamos el cursor encima). Este código se encargará de mostrar u ocultar las distintas opciones de menú, mediante los estilos de CSS.

➔ **Opción 2**: para la segunda opción el trabajo del código JavaScript es bastante más complejo:

- ✔ 1. En este caso, la página HTML sólo contendrá los menús principales.
- ✔ 2. En el **CSS** definimos unos cuantos estilos, correspondientes a los distintos niveles del menú.
- ✔ 3. En el archivo **JavaScript**, declaramos una serie de **arrays**, que contendrán tanto el título como los enlaces de los menús.
- ✔ 4. A partir de ahí y mediante la inserción de **código HTML** desde JavaScript (con innerHTML), iremos creando los distintos div que contendrán las subsecciones.
- ✔ 5. Finalmente aplicamos el estilo a los menús, que habíamos definido en el archivo **CSS**.

En los recursos para ampliar, puedes ver estos dos procesos mucho más detalladamente.

Vídeo: menús con JavaScript

Visualiza este vídeo en el que podrás ver varios menúes hechos con JavaScript.
https://www.youtube.com/embed/-QtEhkUUEA4

9. GESTIÓN DE LA APARIENCIA DE LA VENTANA. MODIFICACIÓN DEL ASPECTO DEL NAVEGADOR Y DOCUMENTO

Una parte esencial de las páginas web son las **ventanas**. A través de ellas podemos definir el **aspecto** de nuestra página. Esto puede influir dependiendo del navegador que estemos utilizando, con lo cual podemos definir también, de alguna manera, el aspecto de éste.

A la hora de abrir nuevas páginas, se pueden abrir en una **nueva pestaña** del navegador, o en una **nueva ventana**, lo cual nos permite cambiar su **apariencia** mediante JavaScript.

PRESTA ATENCIÓN

Es importante remarcar que las funcionalidades que vamos a presentar sólo funcionarán con ventanas que hayan sido creadas por nuestro propio programa de JavaScript. No funcionará para ventanas que haya abierto el usuario por sí mismo.

Esto tiene que ver con las **restricciones** que vimos en unidades anteriores, para evitar posibles daños causados por un programa de script.

- Así pues, en el momento de **crear una ventana** desde JavaScript (veremos cómo hacerlo en el siguiente capítulo), podremos, por ejemplo:
  - ✔ Definir el **contenido** de esa ventana. Puede ser una página HTML creada por nosotros, o cualquier otra URL externa.
  - ✔ Definir el **tamaño**, tanto vertical como horizontal.
  - ✔ Definir la **posición** de la ventana, empezando por la esquina superior izquierda (0,0).
  - ✔ Indicar si la ventana tendrá **barras de desplazamiento** o no.

- ✔ Especificar **márgenes** respecto a los límites de la pantalla (arriba, abajo, izquierda o derecha).
- ✔ Definir el **título** de la ventana.
- ✔ Mostrar o no la **barra de estado**.

➔ Una vez creada la ventana también podremos **efectuar operaciones** sobre esta, como cambiar el tamaño, definir la posición, o poner y quitar el foco sobre la ventana.

```
// Abrimos la ventana con un cierto tamaño
let ventanaGoogle =
window.open("http://www.google.com","Google","width=300,height=3
00")


// Cambiamos el tamaño de la ventana
ventanaGoogle.resizeTo(600, 400)


// Movemos la ventana
ventanaGoogle.moveTo(0,0)


// Damos el foco a la ventana
ventanaGoogle.focus()


// Quitamos el foco a la ventana
ventanaGoogle.blur()
```

10. CREACIÓN DE NUEVAS VENTANAS. COMUNICACIÓN ENTRE VENTANAS

En el capítulo anterior hemos visto las opciones que tenemos cuando creamos ventanas nuevas. Vamos a ver ahora cómo podemos **crear esas ventanas desde JavaScript**.

- **Crear nueva ventana**: para crear una ventana usaremos el **objeto window**, que representa la ventana en la que estamos trabajando. Tendremos que llamar al método open, que nos permite especificar una serie de parámetros.

  Los más utilizados son:

  - La **URL**, o contenido de la página. Es opcional, si no lo indicamos se abrirá una ventana en blanco.
  - El **nombre** de la ventana, también opcional.
  - Una serie de **opciones**, también opcionales, entre los que podemos especificar:
    - **Tamaño**, en píxeles de alto y ancho.
    - **Posición**, en x e y respecto al origen de coordenadas, en la esquina superior izquierda.
    - Si es o no **redimensionable**.
    - Si tiene o no **barras de desplazamiento**.

➔ **Redimensionar**: así, por ejemplo, la siguiente instrucción nos crea una ventana, con la página inicial de Google, un tamaño de 300x300 píxeles, sin barras de desplazamiento, y que no es redimensionable.

```
var ventanaGoogle = window.open("http://www.google.com",
"Google", "width=300,height=300,scrollbars=NO,resizable=NO")
```

Esa ventana que acabamos de crear, además, la guardamos en una variable llamada **ventanaGoogle**, lo que nos permitirá efectuar operaciones sobre ella más adelante, como **redimensionar** o **reubicar**.

➔ **Comunicación entre ventanas**: además, otra opción interesante al crear ventanas desde JavaScript es poder **comunicarnos** entre ellas. Más adelante en el curso veremos cómo crear formularios, que sería una forma más avanzada de comunicación, pero veamos ahora opciones más básicas.

```
// Abrimos una ventana en blanco
var myWindow = window.open("", "MsgWindow",
"width=200,height=100");


// Escribimos un texto en esa ventana
myWindow.document.write("<p>This is 'MsgWindow'. I am 200px wide
and 100px tall!</p>");


// Cerramos la ventana anterior, y abrimos otra con la página de
Google
myWindow.close();
myWindow = window.open("https://www.google.com");
```

En el ejemplo vemos como abrimos una ventana, y nos comunicamos con ella, primero para escribir un cierto texto, y después para cerrarla y abrirla otra vez con una URL diferente.

11. COOKIES. DEFINICIÓN, CREACIÓN, ALMACENAMIENTO Y RECUPERACIÓN

A menudo, cuando visitamos una página web a la que ya habíamos accedido alguna vez, nos encontramos con que la página recuerda cierta **información** que ya habíamos introducido en su día.

A continuación, veremos **qué es una cookie** y **aspectos relevantes sobre su utilización**:

- **Concepto de cookie**:

Esto se debe al uso de cookies, que es la herramienta que permite que una página web guarde información en la máquina del cliente para ahorrar tiempo a éste, cada vez que visita la misma web.

- **Utilización**: la única información que se guarda en una **cookie** es una **cadena de texto**, asociada a una cierta **clave**, que podrá ser leída con cada petición al servidor.

  En los últimos tiempos se ha generado cierta polémica por la utilización de cookies y la posible pérdida de **privacidad** que esto pueda provocar en el usuario.A raíz de esto han salido **nuevas normativas de la LOPD** que regulan el uso de las cookies, y los avisos que se tienen que ofrecer al usuario para su uso.

En posteriores unidades veremos más detalles acerca del uso de cookies en JavaScript, como su creación, lectura y escritura.

Vídeo: ¿son las cookies ilegales si no pido consentimiento?

Visualiza este vídeo para ver aspectos relacionados con la legalidad de las cookies.
https://www.youtube.com/embed/k536jzcsUcs

Actividad de aprendizaje 2: uso de cookies

Siguiendo con la idea de los colores corporativos, nos proponen ahora una forma de agilizar el trabajo del usuario.

- Mediante el uso de cookies tendremos que recordar el color escogido por el usuario, de entre 4 disponibles.
- Además, al clicar un botón, se abrirá una nueva ventana, indicando el color escogido.

Ponte de acuerdo con un compañero para realizar esta actividad y entregad el archivo .html en el espacio habilitado para ello.

Cuestionario

Lee el enunciado e indica la opción correcta:

¿Cómo accedemos a la venta padre?

a. window.parent.

b. rwindow.top.

c. window.root.

Lee el enunciado e indica la opción correcta:

¿Para indicar la fecha de caducidad de una cookie usaremos?

a. date.

b. max-date.

c. max-age.

Lee el enunciado e indica la opción correcta:

¿Qué método es útil para dividir una cadena en otras más pequeñas?

a. concat().

b. split().

c. charAt().

RESUMEN

Esta unidad nos ha servido para seguir avanzando en algunos aspectos de **JavaScript**, más allá de los conocimientos básicos de la unidad anterior.

En concreto hemos empezado a trabajar con los **objetos del lenguaje**, especialmente los predefinidos o nativos de JavaScript, como por ejemplo Date o **Array**.

Hemos visto la diferencia entre éstos y otro tipo de objetos que nos permiten la interacción entre el usuario y el navegador, como son **window** o **document**.

Hemos visto como generar **código HTML** desde JavaScript y poder acceder a sus elementos, para modificar el contenido de una página web.

Respecto a los **marcos**, hemos conocido un poco más de qué se trata esta tecnología, los usos y ventajas que tiene y como trabajarlos con JavaScript. Pudiendo incluso comunicarnos entre diferentes marcos de una misma página.

Además, hemos visto otros temas relacionados con el aspecto de las **ventanas**, los **menús** o la **comunicación entre ventanas**.

Finalmente, hemos hecho un repaso de qué son las **cookies** y cómo podemos usarlas en JavaScript.

RECURSOS PARA AMPLIAR

PÁGINAS WEB

- Ejemplos de menú desplegable con javascript (I): *https://aprende-web.net/jspracticas/desplegable/desplegable1.php* [Consulta junio 2022].
- Ejemplos de menú desplegable con javascript (II): *https://aprende-web.net/jspracticas/desplegable/desplegable2.php* [Consulta junio 2022].
- Frameset - HTML: Lenguaje de etiquetas de hipertexto | MDN: *https://developer.mozilla.org/es/docs/Web/HTML/Element/frameset* [Consulta junio 2022].
- Html frame - Buscar con Google: *https://www.google.es/search?q=html+frame&bih=1785&biw=1120&hl=es&ei=V8uUYoXWFcynlwSPkZ-YBQ&ved=0ahUKEwjF3fbdrof4AhXM04UKHY_IB1MQ4dUDCA0&uact=5&oq=html+frame&gs_lcp=Cgdnd3Mtd2l6EAMyBAgAEEMyBQgAEIAEMgUIABCABDIFCAAQgAQyBQgAEIAEMgUIABCABDIFCAAQgAQyBQgAEIAEOgcIABBHELADOgcIABCwAxBDOgYIABAeEBZKBAhBGABKBAhGGABQzQZY1whggwxoAXAAeACAAXeIAYsCkgEDMi4xmAEAoAEByAEKwAEB&sclient=gws-wiz* [Consulta junio 2022].

BIBLIOGRAFÍA

PÁGINAS WEB

- JavaScript and HTML DOM Reference: *https://www.w3schools.com/jsref/default.asp* [Consulta junio 2022].
- JavaScript Standard Style: *https://standardjs.com/rules.html* [Consulta junio 2022].
- JSON - JavaScript | MDN: *https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/JSON* [Consulta junio 2022].
- Página de W3C: *https://www.w3.org/* [Consulta junio 2022].

GLOSARIO

- **BOM**: browser Object Model, son los objetos que tienen relación con el navegador o la pantalla.
- **Código anidado**: instrucciones dentro de otras instrucciones del mismo tipo.
- **Div**: etiqueta HTML que permite agrupar varios elementos de bloque.
- **DOM**: document Object Model, son los objetos que tiene relación con el documento o contenido de la página.