



PRUEBAS DE FUNCIONALIDADES Y OPTIMIZACIÓN DE PÁGINAS WEB (UF1306)

Familia profesional: Informática y Comunicaciones

[Manual de contenidos](#)

Reservados todos los derechos. El contenido de esta obra está protegido por la ley, que establece penas de prisión o multas, además de las correspondientes indemnizaciones por daños y perjuicios, para quienes reprodujeren, plagiaren, distribuyeren o comunicaren públicamente en todo o en parte, una obra literaria, artística o científica, o su transformación, interpretación o ejecución artística fijada en cualquier tipo de soporte o comunicada a través de cualquier medio, sin la preceptiva autorización.

Todos los nombres propios de programas, sistemas operativos, equipos, hardware, programas de afiliación, páginas web, etc. que aparecen en esta publicación son marcas registradas de sus respectivas compañías, organizaciones y propietarios y tan solo se muestran a modo informativo.

Conzepto Comunicación Creativa

ÍNDICE

1. VALIDACIONES DE DATOS EN PÁGINAS WEB

- 1.1. Funciones de validación.
 - 1.1.1. Descripción de las funciones.
 - 1.1.2. Utilidad de las funciones.
 - 1.1.3. Implementación de las funciones.– Validaciones alfabéticas, numéricas y de fecha.
 - 1.1.4. Definición de validaciones.
 - 1.1.5. Código de validación.
 - 1.1.6. Ejecución del código de validación.
- 1.2. Verificar formularios.
 - 1.2.1. Identificación de datos.– Implementación del código de verificación.
 - 1.2.2. Comprobación de los datos introducidos por el usuario.

2. EFECTOS ESPECIALES EN PÁGINAS WEB

- 2.1. Trabajar con imágenes: imágenes de sustitución e imágenes múltiples.
 - 2.1.1. Selección de imágenes.
 - 2.1.2. Optimización de imágenes.
 - 2.1.3. Implementación de código con varias imágenes.
- 2.2. Trabajar con textos: efectos estéticos y de movimiento.
 - 2.2.1. Creación de textos mejorados y con movimiento.
 - 2.2.2. Implementación de efectos.
 - 2.2.3. Adecuación de los efectos a la página web.
- 2.3. Trabajar con marcos.
 - 2.3.1. Dónde utilizar los marcos.
 - 2.3.2. Limitaciones de los marcos.
 - 2.3.3. Alternativas a los marcos.
- 2.4. Trabajar con ventanas.
 - 2.4.1. Creación de varias ventanas.
 - 2.4.2. Interactividad entre varias ventanas.
- 2.5. Otros efectos.
 - 2.5.1. Efectos con HTML.
 - 2.5.2. Efectos con CSS.
 - 2.5.3. Efectos con capas.

3. PRUEBAS Y VERIFICACIÓN EN PÁGINAS WEB

- 3.1. Técnicas de verificación.
 - 3.1.1. Fundamentales.
 - 3.1.2. Técnicas HTML.
 - 3.1.3. Técnicas CSS.
- 3.2. Herramientas de depuración para distintos navegadores.
 - 3.2.1. Utilidades para HTML.
 - 3.2.2. Utilidades para javascripts. cve: BOE-A-2011-19503
 - 3.2.3. Utilidades para CSS.

- 3.2.4. Utilidades para DOM.
- 3.3. Verificación de la compatibilidad de *scripts*.
 - 3.3.1. Parámetros para distintos navegadores.
 - 3.3.2. Creación de código alternativo para diversos navegadores

Conzepto Comunicación Creativa

OBJETIVOS

OBJETIVO GENERAL

Crear y publicar páginas web que integren textos, imágenes y otros elementos, utilizando lenguajes de marcas y editores apropiados, según especificaciones y condiciones de "usabilidad" dadas y realizar los procedimientos de instalación y verificación de las mismas en el servidor correspondiente.

OBJETIVOS ESPECÍFICOS

C1: Aplicar técnicas de prueba y verificación de la integración de los componentes en la página web para comprobar parámetros de funcionalidad y «usabilidad», de acuerdo a unas especificaciones recibidas. cve: BOE-A-2011-19503

CE1.1 Identificar las fases que intervienen en la verificación de la integración de componentes en páginas.

CE1.2 Clasificar los distintos tipos de archivos que se van a integrar en la página, verificando la instalación del *plug-in* correspondiente en el navegador web.

CE1.3 Verificar la integración de *scripts* ya desarrollados en páginas web para probar su funcionalidad:– Seleccionar varios navegadores.

- Definir los entornos de prueba.
- Identificar los parámetros a verificar.
- Documentar los procesos realizados

CONTENIDOS

1. VALIDACIONES DE DATOS EN PÁGINAS WEB

En el campo de la informática, el proceso de validación de datos es uno de los factores más importantes a tener en cuenta, tanto en la integridad de los datos como en el campo de la seguridad. Validar datos es controlar y testear cada uno de los datos de entrada de un sistema antes de aceptarlos como válidos. No debe confundir el término validar con el término verificar, como verá más adelante. Este proceso es especialmente importante en sistemas que se encuentran conectados a redes de datos como, por ejemplo, Internet.

Validar los datos es una cuestión de seguridad y de integridad de los datos. La vulnerabilidad en la web es fruto de la falta de validación apropiada de las entradas del usuario o del entorno de trabajo. Para evitar problemas como la inyección de código o el ataque de traspaso de directorios es necesario seguir una serie de pautas preventivas. Nunca se debe confiar en los datos introducidos por el usuario para garantizar, en la medida de lo posible, la robustez y seguridad del sistema. Un ejemplo común de una aplicación vulnerable es:

```
<?PHP  
  
$template="red.PHP";  
  
If(isset ($_COOKIE['TEMPLATE']))  
  
$template=$_COOKIE['TEMPLATE'];  
  
Include ("{/home/users/PHPguru/templates/" . $template);  
  
?>
```

Para destruir la seguridad de este sistema, tan solo, debería mandar la siguiente petición de HTTP:

GET /vulnerable.PHP HTTP/1.0

Cookie: TEMPLATE=../../../../etc/passwd

La cadena “../../../../” permite acceder al directorio raíz de este al directorio de contraseñas de UNIX.

En el ejemplo anterior la falta de validación en el lado del cliente en el momento leer una cookie de usuario, pone en riesgo las contraseñas del sistema en el lado del servidor. La validación de los datos de entrada se realiza en el lado del cliente mediante lenguajes de *script* como, por ejemplo, JavaScript y también en el lado del servidor mediante código PHP o ASP. En aplicaciones críticas son necesarias ambas validaciones, ya que, simplemente con desactivar JavaScript del navegador se hace posible enviar datos intencionadamente erróneos para que sean procesados por el lado del servidor.

Es imprescindible comprobar que los datos introducidos vía URL o a través de formularios son los esperados. Por ejemplo, una llamada a un fichero PHP con el siguiente formato: “documento.php?id=12”, envía el parámetro id con valor 12. Eso mostraría los datos del documento cuyo código coincide con dicho valor numérico. El código PHP debe verificar que el dato tras el signo igual “=” es siempre un número; en caso de no hacer, un atacante podría cambiar dicho número por código SQL malintencionado y borrar, por ejemplo, la base de datos de la aplicación web.

Esto es posible debido a que el SQL no discrimina en los datos de entrada la existencia de instrucciones de control como AND, SELECT o incluso EXEC. Si no se verifica la correcta sintaxis de dichos datos, es posible que una línea de código SQL del tipo “SELECT id, nombre FROM productos where id='\$entrada_de_usuario'”, se convierta en “SELECT id, nombre FROM productos where id='1' EXEC master.dbo...”, ejecutando un comando de sistema operativo...

Los riesgos asociados a una deficiente validación de datos pueden estar asociados a la integridad de los mismos, al formato erróneo de los datos de entrada o al incumplimiento de las reglas de negocio. Por ejemplo, se vulnera la integridad de los datos cuando un usuario introduce de manera malintencionada datos erróneos. También, se vulnera el formato de los datos cuando el usuario introduce los datos utilizando un formato incorrecto o fuera de rango y estos provocan un fallo en la aplicación. Y, por último, cuando se incumplen las reglas de negocio se puede producir un comportamiento no previsto en la ejecución de la aplicación.

1.1. Funciones de validación

Para saber qué se puede esperar de las funciones de validación, hay que entender que está ante una validación de tipo formal. No se menciona nada de la veracidad de la información que está siendo validada por la batería de funciones de validación. Se trata de un filtro previo que se puede y se debe realizar como mínimo en el lado del servidor, por lo que las funciones que se implementan en este tema harán uso necesariamente de algún tipo de lenguaje de servidor. Aunque, se centre en los ejemplos con JavaScript. Esto tiene sentido, ya que, aunque sea imprescindible realizar la validación del lado del cliente, sí que es conveniente hacerlo, puesto que, uno de los objetivos de la validación es perfeccionar la experiencia de usuario y evitar que se cometan errores a la hora de interactuar con la interfaz de usuario o UI, siglas en inglés que significan *User Interface*.

Mediante funciones de validación es posible evitar toda una serie de vulnerabilidades y conseguir que los datos de entrada mantengan una integridad y coherencia aceptables. Dicho a grandes rasgos, debe seguir las siguientes directrices de seguridad:

- Evitar el almacenamiento de datos sensibles en campos ocultos, y si esto no es posible, los almacenará cifrados.
- Controlar el tipo de los datos introducidos (numérico, alfabético, alfanumérico, etc).
- Manejar el rango de valores permitidos en los datos de entrada.
- Controlar la longitud máxima válida en caso de entradas alfanuméricas.
- Verificar que el dato resultante de una operación realizada con los datos de entrada está dentro del rango de valores permitido antes de asignarlo a la siguiente variable.

- Bloquear la posible inyección de código SQL malicioso eliminando de las cadenas alfanuméricas de entrada determinados caracteres ASCII. Por ejemplo '=*?'. El uso de URLs amigables puede evitar la necesidad de pasar parámetros vía URL.
- En el caso de servidores web compartidos debe evitar la exposición del código fuente y almacenar los datos importantes en la base de datos en lugar de almacenarlos en variables.

Por ejemplo, el código ISBN (*International Standard Book Number*) es un identificador único para libros desde 1970 a nivel mundial. Este identificador fue creado por una cadena de librerías británicas en 1966 y su uso se extendió posteriormente a todo el país. Originalmente, constaba de 10 dígitos, pero a comienzos de siglo XXI era tal la cantidad de libros impresos, que se preveía que el número de dígitos fueran suficientes no fuesen suficientes y en 2007 se amplió a 13 dígitos.

Por otro lado, los ISBN de 10 dígitos también se ampliaron con el prefijo 978. Cuando se agoten los ISBN antiguos, se empezará a utilizar el prefijo 979. Además, los ISBN antiguos tenían un dígito de control para evitar errores que debe ser recalculado por el nuevo sistema. Para acabar de complicar la tarea de escribir una función de validación para códigos ISBN, algunas editoriales decidieron no utilizarlos y, en su lugar, usar el EAN13 (códigos de barras usados en los productos de venta en general). El código de barras que aparece en las contraportadas de los libros está en formato EAN13, a veces también acompañado de un código Bookland de 5 dígitos que indica el país de origen y el PVP recomendado. El número que codifica el código de barras principal comienza por 978 en el caso de los libros. Existen otros códigos similares al ISBN, como el ISSN para las revistas y periódicos, el IBSN para los blogs, o el ISAN para las producciones audiovisuales.

Una función de validación de códigos ISBN permitiría evitar errores de entrada a la hora de buscar un libro en una base de datos, pero, aunque el código ISBN sea válido, siempre hay una mínima posibilidad de que el libro no exista, o que al menos, no esté en una base de datos. En este caso, el dato de entrada sería válido, pero sin embargo no pasaría el proceso de verificación.

1.1.1. Descripción de las funciones

Una función de validación es aquella que recibe un valor o conjunto de valores, introducidos por el usuario, a fin de comprobar si cumplen con una serie de restricciones impuestas por la aplicación. Existen infinidad de comprobaciones posibles. Algunas de las más habituales son, por ejemplo, comprobar que se ha cumplimentado un campo de formulario obligatorio o, que se ha elegido un valor de una lista desplegable, verificar que el nombre y los apellidos del usuario solo contienen caracteres alfabéticos o comprobar si la dirección de correo electrónico es válida. Puede clasificar el tipo de funciones de validación dependiendo de qué tipo de comprobación realizan, tal como puede ver en el siguiente ejemplo:

- **Entrada obligatoria.** Evita que el usuario deje un campo vacío.
- **Comparación con un valor determinado.** Valida que el dato de entrada sea de un determinado tipo, o bien, que cumpla una condición con respecto a un valor dado, es decir, que sea menor, mayor, menor o igual, mayor o igual.

- **Comprobación con respecto a un intervalo.** Valida que el dato de entrada esté comprendido en un rango de valores dado. Sería aplicable a datos como la edad, número de años de experiencia en un trabajo anterior, rangos de fechas, etc.
- **Comprobación de patrón.** Valida que el dato de entrada se corresponde con un determinado patrón definido por una expresión regular. Permite comprobar la validez de correos electrónicos, es decir, que una cadena de texto sea un correo electrónico válido. También, permite comprobar fechas haciendo que una cadena de texto tenga un formato de fecha válido.

Además, las validaciones pueden ser simples, por ejemplo, cuando se analiza el contenido de un solo campo; o compuestas, en el caso de que la validación de un determinado campo dependa del contenido de otros campos. Un ejemplo de validación compuesta sería en un formulario de registro de nuevos usuarios de una web, la contraseña es válida si ha sido introducida tanto en el campo contraseña, como en el campo repetir contraseña y ambos campos coinciden.

Otra validación compuesta bastante típica, es la que puede ver en cualquier sistema de búsqueda de vuelos por Internet. Suponga que tiene un formulario de entrada de datos en el que piden los siguientes datos: ciudad o aeropuerto de origen, ciudad o aeropuerto de destino, fecha de inicio del viaje, fecha de regreso y número de personas que viajarán. Si el formulario no restringiera los datos que el usuario puede introducir, podría ocurrir que eligiera dos ciudades, una de origen y otra de destino, que no estuviesen conectadas por vuelos regulares. Podría introducir un nombre de ciudad sin aeropuerto, o inexistente. También, podría ocurrir que la fecha de salida o la de regreso fueran incorrectas o bien que la de regreso fuera anterior a la de salida. Entonces, si el usuario seleccionara como ciudad de origen Málaga, esto le restringiría al usuario las ciudades de destino válidas, de igual forma que la fecha de inicio del viaje le condicionaría y restringiría las fechas válidas en la fecha de regreso.

Por otra parte, lo deseable sería que el sistema de entrada de datos estuviese ideado de forma que los errores se redujesen al mínimo, maximizando además las posibilidades de búsqueda. En algunas páginas de búsqueda de vuelos, se permite, por ejemplo, especificar solo la fecha de salida, el número de días y el presupuesto máximo, de forma que el sistema muestra los destinos que cumplen dicho criterio. La descripción de las funciones de validación es parte del diseño de la interfaz de usuario. No hay una única forma de hacer las cosas.



Más Info



RECURSO MULTIMEDIA



1.1.2. Utilidad de las funciones

Para que las funciones de validación sean útiles, deben cumplir su objetivo según varios criterios. En primer lugar, que cumplan su objetivo en lo relativo a la seguridad y fiabilidad de los datos de entrada, ya que de ello depende la integridad y la consistencia del sistema informático. Se entiende como consistencia que el sistema se comporte de la manera en que se espera que lo haga.

Por otra parte, se entiende como integridad que se garantice que dichos datos sean modificados, creados o borrados solo por usuarios autorizados.

En sentido estricto, ninguna validación del lado del cliente es capaz de garantizar al cien por cien estos criterios de seguridad, ya que es posible falsear los datos de entrada y saltarse las validaciones, porque al residir el código fuente en el lado del cliente es posible alterarlo o desactivarlo. Para conseguir que las funciones de validación sean útiles, desde el punto de vista de la seguridad es imprescindible la validación del lado del servidor.

Sin embargo, cuando los datos de entrada no son críticos para la seguridad del sistema, puede ser suficiente una validación del lado del cliente, siempre que tanto la consistencia como la integridad del mismo no se vean comprometidas. Esto no significa que sea deseable prescindir de la validación del lado del cliente, ya que, existe otro criterio a tener en cuenta en lo relativo a la validación de los datos de entrada, el de la usabilidad.

Las validaciones del lado del cliente deben de estar enfocadas a proporcionar una experiencia de usuario satisfactoria y no intrusiva, donde se encuentre el equilibrio entre dos enfoques contrapuestos. Uno que afirma que el sistema es el que debe controlar al usuario y otro que, por el contrario, mantiene que es el usuario el que debe de tener el control sobre el sistema.

Las funciones de validación tendrán que implementarse teniendo en cuenta estas cuestiones. En qué momento se realizará la validación, cuándo hará el usuario clic en el botón Enviar, a qué nivel de campo de formulario, cuándo se modificarán las entradas. No es conveniente que el usuario se vea constantemente interrumpido con mensajes de error mientras rellena un formulario, ya que, resultaría molesto. Sin embargo, no es la única forma de hacerlo, se puede mostrar una marca de validación en verde indicando que un campo es correcto en lugar de mostrar una ventana contextual alertando de un error. Es una validación a nivel de campo de formulario, pero no intrusiva.

Como verá más adelante, se pueden combinar las funciones de validación y verificación incluso realizándolas siempre en el lado del servidor, pero de forma que no se vea comprometida con la usabilidad. El usuario ni siquiera tiene que enterarse de que se están validando los datos en el servidor. La mejor validación es la que no se ve. En resumen, las funciones de validación serán útiles si cumplen los siguientes criterios:

- Cumplen los requisitos de seguridad y fiabilidad establecidos en el sistema informático.
- Cumplen los requisitos establecidos en cuanto a la usabilidad.

 Más Info

RECURSO MULTIMEDIA



1.1.3. Implementación de las funciones

La validación de los datos de entrada es realizada por funciones cuya estructura en pseudocódigo es la siguiente:

```
FUNCIÓN validación (dato1, dato2, ...) {
```

```
SI (condición que debe cumplir el primer dato de entrada) {
```

```
//Si no se cumple la condición
```

```
MENSAJE DE ERROR ('[CÓDIGO DE ERROR] EL DATO 1 debe cumplir la CONDICIÓN 1.');
```

```
RETORNAR (FALSO)
```

```
//Se finaliza la ejecución de la función
```

```
}
```

```
SI (condición que debe cumplir el segundo dato de entrada) {
```

```
//Si no se cumple la condición
```

MENSAJE DE ERROR ('[CÓDIGO DE ERROR] EL DATO 2 debe cumplir la CONDICIÓN 2.');

RETORNAR (FALSO)

//Se finaliza la ejecución de la función

}

//Si la ejecución de la función ha llegado hasta aquí, es que se han cumplido todas las condiciones exigidas.

RETORNAR (VERDADERO);

}

La estructura lógica anterior verifica una batería de condiciones que debe cumplir el conjunto de datos de entrada, en su totalidad, para ser válido. Si cualquiera de las condiciones no se cumple, la función retorna un valor falso. Solo en el caso de que se cumplan todas y cada una de las condiciones se retorna un valor verdadero. Aunque la estructura anterior es perfectamente válida a nivel lógico, no es conveniente realizar la validación sobre un conjunto de datos muy amplios por motivos de usabilidad. Es mucho mejor que se validen los datos mientras se están introduciendo, siempre que sea posible, para evitar molestias al usuario.

A veces no queda más remedio que enviar un formulario para su validación en el lado del servidor. En este caso siempre se intenta causar la menor molestia, impidiendo que se borren todos los datos del formulario y señalando de alguna manera los errores que se cometieron en la introducción de datos y la posible solución de la manera más amigable posible. Si el usuario introdujo mal una dirección de correo electrónico, se puede marcar en rojo ese campo del formulario y mostrar un mensaje de error que le ayude en la introducción de los datos. Aunque en el siguiente epígrafe verá cómo puede implementar directamente en JavaScript los diferentes tipos de funciones de validación numéricas, alfanuméricas, de fecha, de email, etc; en la práctica, lo más habitual y cómodo es utilizar librerías escritas para facilitar dicha tarea, bien para utilizar por separado o como un *plug-in* de validación escrito para una biblioteca JavaScript.

A la hora de elegir qué librería se utiliza para validar los formularios o si no usa ninguna, no solo debe tener en cuenta que funcione bien a nivel técnico y haga lo que tiene que hacer. Esto es, por supuesto, un pre requisito; pero la otra mitad del problema es no permitir que la validación de entrada se convierta en una mala experiencia de usuario. No es solo que esto pueda pasar, sino que es una de las principales deficiencias de la interfaz de usuario en las páginas web, por lo que hay que elegir bien.

Factores a la hora de elegir qué *script* o librería de validación utilizar, serían lo extendido que está su uso; es decir, si tiene actualizaciones, soporte técnico, etc. la claridad de la documentación existente, la facilidad de integración con su proyecto, dependencias de otras librerías, amplitud de casos contemplados, es decir, cómo de completa es la solución en términos de diferentes tipos de

validación de formulario y, quizás, el segundo factor más importante, el impacto de la librería de validación sobre la experiencia final del usuario.

- **Plug-ins de validación para JQuery.** Entre la gran cantidad de *plug-ins* de validación de formularios existentes para la biblioteca JQuery, puede destacar Validity, JQuery Ketchup, JQuery Validate, Parsley.js, Formance.js o Verify.js.
- **JQuery Validate.** En primer lugar, tiene JQuery Validate, uno de los más asentados y utilizados *plug-ins* de validación. Cubre prácticamente cualquier situación que pueda necesitar a la hora de validar un formulario, incluso formularios dinámicos o subida de ficheros, permite la creación de contenedores para mostrar errores y es compatible con las versiones 1.6.4, 1.7.2, 1.8.3 y 1.9.0 de JQuery. Depende de esta librería para funcionar, pero si ya la está utilizando de todas formas, no tendrá problemas.
- **Verify.js.** Otra solución dependiente de JQuery es Verify.js. Una librería muy interesante que permite validaciones configurables mediante parámetros, validaciones asíncronas y validaciones de grupo. La API funciona mediante la definición de datos-atributos asignados usando class en cada campo del formulario para indicarle qué es y qué es lo que se debe validar. Por ejemplo:
`<input value="xx" data-validate="number,required">`

De la misma manera, se puede utilizar para agrupar ciertas validaciones que son dependientes entre sí. Además, permite definir reglas de validación asíncronas mediante la definición de *callbacks*. Una regla de validación de este tipo no se valida en el momento, sino cuando se recibe la respuesta de la función definida en el callback tras hacer una llamada a un *script* de servidor.

- **Parsley.js.** Parsley.js es una librería de validación de uso general que promete realizar el proceso de validación de lado del cliente prácticamente sin escribir una sola línea de código JavaScript. Actualmente, Parsley.js ha integrado en su código a Validate.js, con lo que hay cientos de validaciones disponibles y fáciles de utilizar. Se integra muy bien con otros frameworks como Djando o Rails.

Parsley utiliza una API DOM que permite configurar casi todo directamente desde el DOM. El prefijo DOM por defecto de Parsley es data-parsley. Eso significa que si en la configuración vea una propiedad llamada "spain", esta puede ser modificada vía DOM mediante data-parsley-spain="valor". A continuación, verá un ejemplo de validación de un campo email en el que la validación se realiza en el momento en que el contenido del campo se modifica:

```
<label for="email">Email* :</label>
<input type="email" name="email" data-parsley-trigger="change" required/>
```

Un campo de texto donde se requiere un mínimo de 20 caracteres y un máximo de 200 con el correspondiente mensaje de error que se mostrará en caso de que no se pase la validación. En este caso la validación se realiza conforme se escribe data-parsley-trigger="keyup":

```
<label for="comentario">Comentario (mín 20 caracteres, máx 200):</label>
```

```
<textarea name="comentario" data-parsley-trigger="keyup" data-parsley-length="[20,200]" data-parsley-validation-threshold="10" data-parsley-minlength-message="El comentario debe tener al menos 10 caracteres"></textarea>
```

Lo bueno que tiene Parsley es que permite implementar con poco esfuerzo más la validación del lado del servidor mediante Ajax. Para ello, debe utilizar el *plug-in* Parsley Remote.



TOME NOTA

La validación realmente segura es la del servidor, la del cliente es solo recomendable para mejorar la experiencia de usuario, evitar que se envíen formularios mal completados y reducir la carga de trabajo.

AngularJS. AngularJS no es una simple librería de JavaScript, es un framework de código abierto que trae a la programación front-end el paradigma Modelo Vista Controlador o MVC, para estructurar lo que se conoce como aplicaciones web de una sola página. Es la alternativa de Google a BackbonesJS o EmberJS.

La validación de formularios con AngularJS no requiere de ningún *plug-in*, ya que está implementada dentro del framework. Se basa en definiciones de datos-atributos, aunque con otro formato más al estilo HTML5:

```
<input type="text" ng-pattern="[a-zA-Z]"/>  
  
<input type="email" name="email" ng-model="user.email"/>  
  
<input type="number" name="edad" ng-model="user.age"/>
```

Sin embargo, las herramientas de validación incluidas son más bien escasas, con lo que tiene que tirar de la creación de directivas de validación personalizadas haciendo uso de ng-pattern o ui-validate.

- **Mootools.** Mootools es una alternativa, también de código abierto, a JQuery. Para validar formularios en Mootools se puede hacer uso de un componente llamado FormValidator. Para usarlo, necesita inicializar el componente utilizando una serie de parámetros. El primer parámetro es el identificado CSS-id del formulario y luego los parámetros stopOnFailure, usesTitle, errorPrefix, onFormValidate y onElementValidate que permiten decidir cómo se comportará el procedimiento de validación. Por ejemplo, si stopOnFailure es *true* está indicando que desea que en el caso de que un campo del formulario falle la validación, el resto de campos del formulario se desactiven temporalmente. Puede guardar un texto “Error de tipo”, por ejemplo, que se usará como prefijo de los errores de validación asignando el mismo a errorPrefix. Un ejemplo de configuración sería la siguiente:

```
Window.addEvent('domready',function(){  
miValidacion = new Form.Validatos.Inline $('#fRegistro'),{  
stopOnFailure: false,  
usesTitles: true,  
errorPrefix: "Se produjo un error",  
onFormValidate: function(passed,form,e){  
if(passed){  
form.submit();  
}  
}  
});  
});
```

Luego, a nivel de campo de formulario necesita especificar el tipo de validación que implementar numérica, alfanumérica, de email, etc. El número de caracteres mínimo o máximo, etc. Por ejemplo:

```
<input type="text" class="validate-alphanum minLength:3 maxLength:30" name="profile [username]" maxlength="30"/>
```

- **Validación de formularios con librerías independientes de uso general.** Se trata de librerías independientes donde no se necesita usar JQuery, ni Mootools para funcionar. Están ideadas ad-hoc para validar formularios. Son más ligeras y fáciles de usar al estar especializadas en tareas de validación. Algunas de las más interesantes son Validatious y Validate.js.

Validatious es una librería que solo pesa 24 kb y que utiliza el método de asignación de determinadas clases CSS a los campos del formulario para indicar cómo deben ser tratadas a la hora de validarlos. Por ejemplo, si quiere que un campo sea validado como contraseña, campo requerido y con, al menos, 6 caracteres de longitud, solo tendrá que añadir a las clases required, password y min-length_6 la etiqueta input correspondiente:

```
<label for="password">Contraseña</label>  
  
<input type="password" name="password" id=password" class="required password min-length_6"/>
```

El objetivo declarado de Validate.js es proporcionar un sistema de validación que funcione en varios frameworks y lenguajes de programación. Para ello, las restricciones de validación son declaradas en un objeto JSON y compartidas entre los clientes en el servidor. Funciona con Python, Ruby on Rails y node.js.

La definición de restricciones tiene la siguiente estructura:

```
{  
  <attribute>: {  
    <validator name>:<validator options>  
  }  
}
```

Si, por ejemplo, desea validar un número de tarjeta de crédito, define el siguiente objeto de validación, donde se especifica la expresión regular correspondiente:

```
Var constraints = {  
  
  numeroDeTarjeta: {  
    presence: true,  
  
    format: {  
      pattern: /^(34|37|4|5[1-5]).*$/,  
  
      message: "Debe introducir un número de tarjeta válido"  
    },  
  
    Length: {is:16},  
  
    codigoPostal: function(value, atributos, attributeName){  
  
      if(!(/^(34|37).*$/.test(atributos.numeroDeTarjeta)) return null;  
  
      return {  
        presence: true,  
  
        length: {is: 5}
```

```
};  
}  
};
```

Para validar un número de tarjeta de crédito se realizaría la siguiente llamada a la función validate:

```
Validate ({numeroDeTarjeta: "9254545454545454", codigoPostal:"458"}, constraints);
```

Soporta también validaciones asíncronas, útiles por ejemplo para comprobar si el nombre de usuario elegido por el usuario está libre, preguntando al servidor. Para ello, Validate.js utiliza la función validate.async, que tiene la misma sintaxis y acepta las mismas opciones que la función validate.

- **Validación de formularios específicos.** Accounting.js es una librería JavaScript muy pequeña y útil para formatear/validar valores numéricos y de moneda. Permite, opcionalmente, mostrar los valores numéricos en estilo de hoja de cálculo. No requiere ninguna otra librería para funcionar. Por ejemplo, para formatear una cifra como libras esterlinas, se usa la siguiente sintaxis:

```
Accounting.formatMoney(254500, {symbol: "£", format: "%v %s"}); // 254,500.00 £
```

Muy útil para sistemas de facturación online. Mientras el usuario introduce los datos, estos son formateados evitando errores de introducción de datos.

Se resume en una tabla las características principales de cada una de las soluciones que puede utilizar para validar formularios. Cada librería está valorada de 1 a 10. Se tiene en cuenta las dependencias de cada una. Si no depende de ninguna librería la puntuación es 10 en ese apartado:

Librería		Experiencia de uso	Facilidad de desarrollo	Amplitud/Casos contemplados	Dependencia de otras librerías	Puntuación media
jQuery Validation plug-in	9	8	8	6	7,75	
jQuery Ketchup plug-in	8	8	8	8	8	
Validate.js	9	8	6	9	8	
Parsley	9	10	8	8	8,75	
Verify.js	10	6	8	8	8	
gValidator	6	6	6	10	7	
Validity	8,5	8	8	8	8,13	
Angular.js	8	8	7	10	8,25	

1.1.4. Validaciones alfabéticas, numéricas y de fecha

Las validaciones sobre los datos de entrada pueden ser clasificadas en tres grandes grupos:

- Validaciones alfabéticas.
- Validaciones numéricas.
- Validaciones de fecha.

Como ya se ha visto anteriormente, esto no es solo cuestión de integridad de los datos que posteriormente se grabarán en la base de datos, también es una cuestión de seguridad muy importante a fin de evitar inyección de código malintencionado, que pueda resultar en una pérdida de datos o un mal funcionamiento de la aplicación web.

Las validaciones alfabéticas tienen como objetivo verificar aquellos datos de entrada de tipo alfanumérico o puramente alfabético con el fin de que no contengan ningún carácter extraño. Un ejemplo de dato de entrada de tipo alfanumérico sería, por ejemplo, una entrada o post en un blog. Un ejemplo de dato de entrada de tipo alfabético es un nombre, apellidos o nombre de población. Una dirección es, en principio, alfanumérica, a no ser que se fraccione en tipo de vía, nombre de vía, número, planta y puerta. En este caso la validación se fracciona en tantos subcampos como se hayan definido. De primeras, podría parecer sencillo realizar este tipo de validaciones. Un dato alfanumérico es una secuencia de caracteres con la siguiente estructura:

D= {c1 c2 c3 c4 c5 c6 c7 c8 ... cn-1 cn}

Un dato D sería alfanumérico cuando cada uno de los elementos que lo componen sea alfanuméricos. Es necesario definir en la práctica cuál es el conjunto de caracteres válidos. En español, además de usar las letras comprendidas entre a-z y A-Z, utiliza vocales acentuadas y el carácter especial ñ y Ñ. Además, también debe tener en cuenta la cedilla Ç y ç. Si, por ejemplo, está validando un nombre propio, tendrá que tener en cuenta que en otros lenguajes también se usan diferentes tipos de acentos y diéresis:

{à è ï ò û ... â ê î ô û ... ä ë î ö ü}

Tener en cuenta esto es importante, ya que no basta solo con decir que un dato es alfanumérico tan solo porque está comprendido entre la s y la z, o entre la A y la Z, o por ser un número. Lo correcto sería que la función de validación verifique que para cada carácter c(i), donde i=0 hasta n, tomara alguno de los valores que defina como válidos para dicha entrada. En el caso de los nombres y apellidos en español, la lista de valores permitidos es la siguiente:

{abcdefghijklmnñopqrstuvwxyz ABCDEFGHIJKLMNOPQRSTUVWXYZ}

Esto se expresa mediante una expresión regular de la siguiente manera:

[a-zA-Z]+

Más adelante, verá más sobre expresiones regulares, esto permitirá comprobar que esta primera aproximación, realmente, está incompleta.

En el caso de un artículo en un blog o de la descripción de un producto, por ejemplo, habrá que añadir además números y algunos signos de puntuación, aunque no todos. Por ejemplo, se pueden permitir las comillas simples, pero no las comillas dobles. Se puede permitir el uso de la coma, el punto y el punto y coma, pero no la introducción de la barra simple o la barra inversa. Hay que tener esto muy en cuenta porque como ya se ha visto anteriormente, esto se utiliza por motivos de seguridad con la finalidad de evitar la inyección de código. Las validaciones numéricas, por su parte, no pasan únicamente por comprobar si se están usando solo caracteres válidos, aunque también, sino por verificar si el rango de valores de entrada coincide con lo que espera. Por ejemplo, si está verificando la edad de un usuario, debe comprobar en primer lugar si hay algún carácter no permitido, es decir, cualquier cosa diferente a los valores comprendidos entre 0 y 9, y, posteriormente, si el dato está dentro de un rango determinado.

Otro ejemplo de validación numérica es, por ejemplo, el caso de datos que expresen precios, medidas o pesos. En una tienda online es muy importante verificar estos datos para evitar errores, ya que, estos errores podrían repercutir en el comercio de forma económica. Por lo tanto, para este tipo de ejemplo debería aceptar datos numéricos con decimales.

Lo principal sería definir el número de decimales que se aceptarían. En el caso del uso de precios se suelen utilizar dos decimales. Hay que tener en cuenta que la mejor validación es la que no se ve, por lo tanto, lo suyo sería controlar la entrada de datos *on the fly*; es decir, mientras el usuario escribe el dato. Por lo tanto, el resultado de esto será que cuando el usuario introduzca el precio de un producto, se le impedirá escribir en el campo más de dos decimales sin necesidad de mostrar ningún mensaje de error. Este sería el ejemplo de una validación de datos amigable y no intrusiva.

Por otro lado, la validación de fechas presenta varios enfoques. El primero y fundamental es evitar que se introduzcan en el sistema fechas inexistentes. Para ello, lo primero que debe tener en cuenta es si el año introducido es bisiesto o no, puesto que en el caso de que el año sea bisiesto, el mes de febrero tendrá 29 días en vez de 28. Para saber si un año es bisiesto, lo primero que debe conocer es el número de días de cada mes en un año cualquiera. Por ejemplo, si el usuario introduce su fecha de nacimiento, debe comprobar si dicha fecha es correcta y mostrar el correspondiente mensaje de error.

1.1.5. Definición de validaciones

En este epígrafe va a aprender a distinguir entre validaciones y verificaciones. **Validar** un dato significa comprobar si cumple unas determinadas características formales, por lo tanto, no garantiza la veracidad del mismo en cuanto a su contenido real. Por ejemplo, imagine que está accediendo a su cuenta en una tienda online y, para ello, es necesario que proporcione su dirección de email previamente proporcionada en un proceso de registro, para validar esta cuenta de correo habría que suponer que es una cadena alfanumérica con las siguientes características:

- Debe comenzar con una cadena alfanumérica a-z0-9 opcionalmente seguida de una o más cadenas alfanuméricas a-z0-9 separadas por puntos. Además, opcionalmente se puede iniciar la secuencia con un guion bajo.
- Esta cadena debe continuar con una arroba @ obligatoriamente.

- Detrás de la arroba debe de contener otra cadena alfanumérica a-z0-9 que podrá estar opcionalmente seguida de una o más cadenas alfanuméricas separadas por puntos.
- Es obligatorio que tenga un último punto seguido de 2 o 3 letras, por ejemplo, .es, .com, .net, etc.

A continuación, puede ver la expresión regular correspondiente a estos requerimientos:

`^[_a-zA-Z0-9.]+(\.[_a-zA-Z0-9-]+)*@[a-zA-Z0-9-]+\.(a-zA-Z0-9-+)*(\.[a-zA-Z]{2,3})`

Una cadena de caracteres que cumpla con estas características será una dirección de correo válida, aunque esto no significa que esta exista. Por lo tanto, si una cadena de caracteres está validada como correo electrónico, posteriormente, se deberá verificar su existencia. Esto se suele realizar enviando un mensaje a dicha cuenta de correo electrónico que, una vez que el usuario lo haya seguido, verifica tanto la existencia del email como la propiedad del usuario. Este proceso de validación se puede hacer tanto en el lado del cliente como del servidor.

Una validación realizada en el lado del servidor no se debe confundir con una verificación solo por donde se ha realizado, debe tener claro que son dos procesos diferentes. Es posible utilizar una expresión regular para validar un email tanto en JavaScript como en PHP. En ambos casos está realizando una validación, no una verificación. En el primer caso la realizaría en el lado del cliente y en segundo en el lado del servidor.

Normalmente, las validaciones se realizan en el lado del cliente y las verificaciones en el lado del servidor, pero esto no significa que no sea posible hacerlo, al contrario. Por ejemplo, una verificación que normalmente se hace en el lado del servidor es la que realizan los Captchas. **Captcha** significa *Completely Automated Public Turing Test to tell Computers and Humans Apart*, lo que en español sería Test de Turing Público y Automático para distinguir a los Ordenadores de los Humanos. Teóricamente un programa o robot de Internet serían incapaces de pasar este test. Se acepta esta afirmación como cierta, aunque constaría de matices.

Actualmente, los algoritmos de análisis de imágenes de Google han alcanzado tal nivel de sofisticación que pone en tela de juicio que los CAPTCHA basados en imágenes, realmente, sean una implementación del Test de Turing. Como verá más adelante, el CAPTCHA basado en imágenes no es el único tipo de CAPTCHA que puede implementar. De hecho, los CAPTCHA basados en imágenes son, básicamente, una simplificación del concepto de Test de Turing que, en realidad, es mucho más amplio. Existe la posibilidad de realizar CAPTCHAS de muchos tipos, de hecho, en su origen el Test de Turing estaba basado en un chat de texto. Un buen ejemplo de una verificación del lado del cliente sería un CAPTCHA que pidiera al usuario que introdujese el resultado de una operación matemática expresada con texto. Por ejemplo, ochenta y ocho más doce. Teóricamente, si el usuario introdujera como resultado cien, se trataría de un humano y no de un robot. Otro ejemplo sería mostrar una serie de palabras de una categoría e insertar una palabra que pertenezca a otra categoría diferente, por ejemplo, una lista de colores donde inserte una palabra que sea un objeto. En este caso se le pediría al usuario que escogiera la palabra del objeto. Este tipo de ejemplos de CAPTCHA puede encontrarlos, por ejemplo, al iniciar sesión en la página de La Junta de Andalucía. Puede ver la imagen a continuación:

Acceso mediante usuario y contraseña

Usuario (NIF/NIE/CIF): Contraseña:

> ¿Es la primera vez que entras?
> ¿Has olvidado la contraseña?

> Acceso con certificado digital o DNI-e
> Acceso a empresas a través de Contrat@

Rellena una de las siguientes opciones:

A Para realizar el control de acceso puedes escribir los caracteres que aparecen en la siguiente imagen, o bien puedes utilizar el método alternativo (opción B):
GnGvWyH

(Se distingue entre mayúsculas y minúsculas)

B Como método alternativo para el control de acceso, responde a la siguiente pregunta:
¿Cuál es un animal en esta lista? [armario, peine, gafas, tigre]

ENTRAR **LIMPIAR**

1.1.6. Código de validación

En este epígrafe se aprenderán algunos conceptos básicos sobre expresiones regulares que van ser muy útiles para realizar validaciones. Las expresiones regulares son por ellas mismas un lenguaje para la descripción de lenguajes, es decir, una expresión regular define un lenguaje. Por ejemplo, la expresión regular `[0-9]+` define un conjunto de caracteres formados por cadenas numéricas de enteros de uno o más dígitos. Otro ejemplo que puede observar es la expresión `herman[oa]` coincide con las palabras hermano y hermana, pero también coincidiría con las expresiones hermanos y hermanas. Si quisiera descartar las últimas dos expresiones, debería utilizar el símbolo `$`, por lo que quedaría la expresión como `herman[oa]$`. En este ejemplo el metasímbolo `$` indica que la expresión debe estar situada al final de la línea.

Vea, a continuación, otros metasímbolos que serán también muy útiles:

- **Asterisco (*)**. Este símbolo indica que debe aparecer un elemento cero o más veces.
- **Suma (+)**. Este símbolo indica que un determinado elemento debe aparecer 1 o más veces.
- **Punto (.)**. Este símbolo indica que debe haber un solo carácter, excepto retorno de carro.
- **Interrogación (?)**. Indica que el elemento precedente es opcional.
- **{n}**. Este símbolo indica que el elemento precedente debe aparecer n veces.
- **{n,}**. Esto indica que el elemento precedente debe aparecer, al menos, n veces.
- **{n,m}**. El símbolo indica que el elemento precedente debe aparecer entre n y m veces.

- **Tubo (|).** Este símbolo se utiliza como o lógico.
- **Acento circunflejo (^).** Este símbolo indica un comienzo de línea.
- **Símbolo del dólar (\$).** Indica el final de la línea.
- **Corchetes [...].** Indican un conjunto de caracteres admitidos, por ejemplo [oa].
- **Corchetes con acento circunflejo [^...].** Esto indica un conjunto de caracteres no admitido.
- **Guion (-).** Este símbolo es un operador de rango, por ejemplo [0-9].
- **Paréntesis (...).** El paréntesis se utiliza para agrupar elementos, por ejemplo ([a-zñ][A-ZÑ]+)+\.
- **Barra invertida (\).** Este símbolo se utiliza para escapar caracteres.
- **Barra invertida con punto (\.).** Como se ha mencionado anteriormente, la barra invertida se utiliza para escapar caracteres, así que, si quiere buscar un punto(.) debe utilizar la barra invertida para ello, ya que el punto no es un metacaracter, sino un carácter especial.
- **\s.** Este símbolo representa un espacio en blanco.
- **\n.** Éste representa el carácter de fin de línea.
- **\t.** Por último, este símbolo representa el carácter de tabulación.

Ahora que ha visto cómo funcionan los metasímbolos básicos, vea un ejemplo de cómo definiría un apellido o un nombre propio español mediante una expresión regular:

(([A-ZÁÉÍÓÚ][a-zñáéíóú]+)(\s)*)*

A continuación, verá una expresión la cual solo permitiría nombres simples o compuestos, siempre que no comiencen por Ñ e incluyan un guion entre medio como, por ejemplo, García-Pérez. La expresión regular quedaría como se ve a continuación:

(([A-ZÁÉÍÓÚ][a-zñáéíóú]+)(\s|-)*)*

La expresión regular anteriormente mencionada significa un campo que permite cadenas de caracteres que comiencen por mayúsculas, ya estén acentuadas o no, seguidas por minúsculas y separadas de la siguiente cadena por un guion o un espacio en blanco. La primera cadena es obligatoria, pero la segunda es totalmente opcional.

En conclusión, las expresiones regulares constituyen una herramienta muy útil que permitirá realizar diferentes validaciones de los datos de entrada. Puede utilizar expresiones regulares tanto en HTML como en JavaScript.

Uso de expresiones regulares en JavaScript

Para usar expresiones regulares en JavaScript tiene dos opciones. La primera es usar el método `match()` del objeto `string`. Este método busca mediante una expresión regular en una cadena, es decir, un `string` y posteriormente retorna las coincidencias en forma de `array`. En el caso de que no se hayan encontrado coincidencias, devolverá un dato `null`. Este método, por defecto, devolverá la primera coincidencia que haya encontrado, pero si quiere que devuelva más de un valor que

coincidan con la búsqueda, es necesario especificar el parámetro /g. A continuación, vea un ejemplo donde se busca el patrón [mp]adre:

```
Var cadena="El director del colegio quiere hablar con mi madre y con mi padre";
```

```
Var acción=cadena.match(/([mp]adre/g);
```

```
Console.log(acción);
```

Al finalizar la ejecución de este trozo de código, el array acción contendrá los datos madre y padre en su interior.

Por otra parte, el método test() del objeto RegExp se utiliza para ejecutar una búsqueda de una coincidencia entre una expresión regular y una cadena. Este método retorna los valores *true o false*. Vea un ejemplo similar al anterior, pero usando el método test().

```
Var cadena="El director del colegio quiere hablar con mi madre y con mi padre";
```

```
Var patron=new RegExp("[mp]adre");
```

```
Var acción=patron.test(cadena);
```

```
Console.log(acción);
```

En este caso el método test() devolverá como resultado el valor true, ya que, ha encontrado coincidencias entre la expresión regular y la cadena. Es importante saber que el método test() del objeto RegExp es más eficiente en cuanto a velocidad de ejecución que el método match() del objeto string. Por lo tanto, si solo necesita saber si un determinado patrón se cumple o no, se recomienda utilizar el método match(), pero para búsquedas más complejas se recomienda utilizar el método test().

Uso de JavaScript para validar una dirección de correo electrónico

Como ya se ha visto anteriormente, una cadena de texto es válida como dirección de correo electrónico si está compuesta por una cadena que comience por caracteres alfabéticos, seguidos de una o varias cadenas de caracteres alfanuméricos separadas por puntos o guiones medios y, a su vez, seguida por una arroba, otra cadena alfanumérica y un punto seguido por 2 a 6 caracteres alfanuméricos, por ejemplo, .com, .es, .cat, .mobi, .travel, etc. Vea el siguiente ejemplo donde podrá observar cómo se realiza una validación de un email en JavaScript:

```
Function cElectronico(emailUsuario)
```

```
{
```

```
Var patron = "[a-zA-Z0-9_.\\-]+@[a-zA-Z0-9\\-\\.]+\\.[a-zA-Z]{2,6}";
```

```
Var eRegular = new RegExp(patron);

Return eRegular.test(emailUsuario);

}
```

Uso de JavaScript para validar un teléfono fijo

Antes que nada, debe establecer que un teléfono fijo en España tiene 9 dígitos y cuyo primer dígito debe comenzar obligatoriamente por 9 o por 8. Esto es debido a la nueva regulación que hubo en el año 2004. Teniendo en cuenta estos datos, se verá a continuación un ejemplo de una función que retornará *true*, si la entrada de datos cumple dicho criterio, en cambio si no los cumple, retornará *false*. Vea el siguiente ejemplo:

```
Function telFijo (numero)

{

Var patron=/^89]\d{8}\$/;

Var eReg = new RegExp(patron);

Return eReg.test(numero);

}
```

Uso de JavaScript para validar un teléfono móvil

Para crear una expresión regular que valide un número de teléfono móvil hay que tener en cuenta que un número de teléfono móvil para que sea válido deber tener 9 cifras de las cuales el primer dígito debe ser un 6 o un 7. A continuación, va a ver una función que retornará *true* si el número móvil cumple dicho criterio, en caso contrario retornará *false*. Vea el ejemplo:

```
Function telfMovil(numero)

{

Var patron=/^67]\d{8}\$/;

Var eRegular = new RegExp(patron);

Return eRegular.test(numero);
```

}

Uso de JavaScript para validar un número CIF de una empresa

El código CIF es el Código de Identificación Fiscal que usan las empresas, es como un DNI para empresas. Este código debe comenzar por una letra que debe estar dentro de un conjunto de letras válidas, además, este código tiene número pares e impares que deben cumplir una determinada relación con respecto al número central. El código que va a ver a continuación contiene una función que devolverá *true* si el código CIF cumple con los criterios establecidos, en caso contrario devolverá *false*. Vea el siguiente ejemplo:

```
Function codigoCif (código)
{
    Par=0;
    Impar=0;
    letrasValidas="ABCDEFGHIJKLMNPQS";
    primeraLetra=código.charAt(0);
    if(código.length !=9){
        //El código CIF debe tener 9 dígitos
        Return false;
    }
    If (letrasValidas.indexOf(primerLetra.toUpperCase()) == -1)
    {
        //La primera letra del CIF no es válida
        //La letra no coincide con ninguna de las válidas ni en minúsculas, ni en mayúsculas
        Retunr false;
    }
```

//Si el código pasa la primera criba, ahora, verifique que es un CIF válido según el //algoritmo que usa Hacienda

```
For (i=2; i<8; i+=2)

{

    Par=par + parseInt (código.charAt(i));

}

For (i=1; i<9; i+=2)

{

    dobleImpares=2*parseInt(código.charAt(i));

    if(dobleImpares >9)

        dobleImpares = 1+(dobleImpares-10);

    impar = impar + dobleImpares;

}

Parcial= par + impar;

Control (10- (parcial %10));

If (control == 10)

{

    Control=0;

}

If (control!=código.charAt(8))

{

//Si esta verificación última no se cumple, el CIF no es válido

Return false;

}
```

Alert (“El CIF es válido”);

Return true;

}

Uso de JavaScript para validar un número NIF

Un NIF es un Código de Identificación Fiscal, en este caso, para personas. Este código es válido si termina en una letra. Además, la cadena numérica del propio código dividida entre 23 da un resto que coincide con un carácter en concreto de cadena de verificación establecida, por ejemplo, si la parte numérica del NIF dividida entre 23 da un resto de 1, la letra final del NIF deberá ser R. Las letras válidas para este código son el siguiente grupo: TRWAGMYFPDXBNJZSQVHLCKET. En el siguiente ejemplo, la función retornará *true* si la entrada cumple el criterio requerido si no, retornará *false*. Vea el siguiente ejemplo:

Function codigoNif (codigoNif)

{

longitudCodigo = códigoNif.length;

códigoDni = códigoNif.substring (0, longitudCodigo-1);

letraNif = códigoNif.charAt (longitudCodigo-1);

if (!isNaN (letraNif))

{

//El NIF no tiene letra

Return false;

}

Else

{

cVerificacion = “TRWAGMYFPDXBNJZSQVHLCKET”;

posición = códigoDni%23;

```
letra = cVerificacion.substring (posición, posición+1);

if(letra!=letraNif.toUpperCase())

{

    //Si no se cumple la comprobación de seguridad, el NIF no es válido

    Return false;

}

Alert("El NIF es válido");

Return true;

}
```

Uso de JavaScript para validar fechas

Para validar fechas, antes deben cumplir ciertos criterios, es por ello que como paso previo las fechas deben respetar el formato oficial en el caso de España, el cual es DD/MM/AAA. El patrón que se usa para validar fechas que cumplan con el formato oficial es el siguiente:

`^\d{1,2}\/\d{1,2}\/\d{2,4}$`

Tomando el patrón visto anteriormente, las fechas que pasarían como válidas serían como los ejemplos siguientes:

05/12/2002

3/1/1993

9/03/33

Si quiere forzar a que el año deba tener 4 dígitos en vez de 2 o 4, debe modificar la expresión regular de la siguiente manera:

`^\d{1,2}\/\d{1,2}\/\d{4}$`

Este patrón forzará a que el año deba tener 4 dígitos obligatoriamente y, gracias a ello, evitar errores y confusiones.

Lo próximo que debe hacer es validar el mes del año, ya que, su valor debe estar comprendido entre 1 y 12. Además, debe conocer si el año es bisiesto o no para saber si los días del mes son correctos o no. Para saber si un año es bisiesto debes dividir los días del año entre 4 y obtener un resto de 0, por otra parte, debe dividir el año entre 100 y obtener un resto distinto de 0. También, debe tener en cuenta que un año será bisiesto si al dividir el año entre 100 y entre 400 su resto es 0. A continuación, verá el ejemplo con código JavaScript:

Function aBisiesto (year)

{

If (((year%4) == 0) && ((year %100) !=0) || ((year %100) == 0) && (year % 400) == 0)

{

 Return true;

}

Else

{

 Return false;

}

Lo próximo que hará es crear un array que contenga el número de días para cada mes del año, pero hay que tener en cuenta que en los años bisiestos el mes de febrero tiene un día más. Por lo tanto, debe crear el siguiente array:

diasMes = new Array (31, (aBisiesto (year))?29:28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31);

La expresión (aBisiesto (year))?29:28 quiere decir que si el año es bisiesto, adoptará el valor 21 y si no lo es, adoptará el valor 28.

A continuación, va a ver todo el código en conjunto para escribir la función de validación de fechas. Como ha visto anteriormente en otros casos, esta función retornará el valor true si la cadena que se verifica es correcta y cumple con los criterios establecidos, en caso contrario retornará el valor false. En la validación, no se admitirán años por debajo de 1900, ni por encima de 2100. Vea el ejemplo siguiente:

function Fecha (cadenaFecha)

```
{  
  
    var minYear = 1900;  
  
    var maxYear = 2100;  
  
    //Pasamos a validar el formato fecha DD/MM/AAAA  
  
    var separador = "/";  
  
    //Primero se valida que sea correcto a nivel formal  
  
    if (!fechaCorrecta (cadenaFecha))  
  
        return false;  
  
    else  
  
        {  
  
            //Si la fecha está correctamente formada, obtiene los valores  
  
            //numéricos del día, mes y año  
  
            var dia = parseInt(getDia(cadenaFecha, separador));  
  
            var mes = parseInt (getMes(cadenaFecha, separador));  
  
            var year = parseInt (getYear(cadenaFecha));  
  
            //A continuación, obtiene el array de días del mes para ese año  
  
            var diasMes = conjuntoDiasMes (year);  
  
            //Si el año está dentro del rango fijado, el mes está comprendido  
  
            //entre el  
  
            //rango 1 y 12 y el día está entre 1 y el número de días máximo de  
  
            //ese mes
```

```
//contando con que el año sea bisiesto o no, la fecha será totalmente correcta.

if ((year>minYear && year < maxYear) && (mes>0 && mes <=12) && (dia>0 && dia<=diasMes[mes-1])){    return true;}else{    return false;}}
```

}

```
//Esta función comprueba que la cadenaFecha cumple el patrón de fecha establecido con formato DD/MM/AAAA
```

```
function fechaCorrecta (cadenaFecha){    var patron = /^(\d{1,2})\|(\d{1,2})\|(\d{4})$/;    var eRegFechaCorrecta = new RegExp(patron);    return eRegFechaCorrecta.test(cadenaFecha);}
```

}

```
//A continuación, obtendrá el fragmento de cadena que contiene el día.
```

```
function getDia (cadenaFecha, separador)
{
    return (cadenaFecha.split(separador)[0]);
}

//A continuación, obtendrá el fragmento de cadena que contiene el mes.

function getMes (cadenaFecha, separador)
{
    return (cadenaFecha.split(separador)[1]);
}

//Ahora, obtendrá el fragmento de cadena que contiene el año.

function getYear (cadenaFecha)
{
    return cadenaFecha.split[2];
}

//La siguiente función retornará un valor true si se trata de un año bisiesto, si no retornará //un valor false.

function aBisiesto (year)
{
    if (((year%4) == 0) && ((year%100) != 0) || ((year%100) == 0) && (year%400) == 0)
    {
        return true;
    }
    else
    {

```

```
return false;  
}  
}  
  
//La función siguiente retorna un array con el número de días de cada año, teniendo en cuenta si es bisiesto o no.  
  
function conjuntoDíasMes(year)  
{  
    return (new Array (31, (aBisiesto(year))?29:28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31));  
}
```

Uso de otras expresiones regulares útiles

A continuación, vea cuál es la expresión regular para validar una entrada de texto como URL. Para ello va a fijarse en el siguiente código:

```
^(ht|f)tp(s?)\:\:\/\/[0-9a-zA-Z]([-\w]*[0-9a-zA-Z])*(:(0-9)*)*(V?)(([a-zA-Z0-9\-\.\?\.\'\\\\\\+\%\$\#_])*)?$/
```

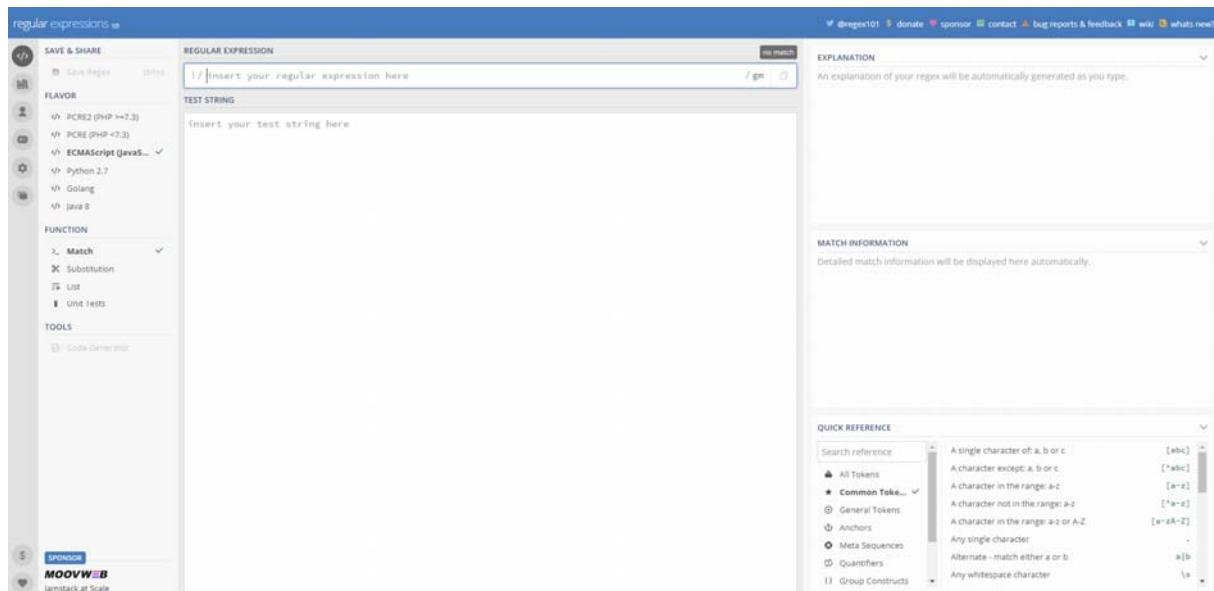
Por otro lado, para validar tarjetas de crédito puede utilizar la siguiente expresión regular que, como puede recordar, solo garantiza que es un número válido, pero no garantiza que ese número exista, ya que, esa acción forma parte del proceso de verificación y no del de validación. Vea el siguiente código de validación:

```
^(67\d{2})|(4\d{3})|(5[1-5]\d{2})|(6011))(-?\s?\d{4}){3}|(3[4,7])\d{2}-?\s?\d{6}-?\s?\d{5}$
```

Por último, va a ver cómo puede controlar la seguridad de una contraseña mediante una expresión regular que fije un formato mínimo válido. Por ejemplo, va a hacer que solo se admitan contraseñas de al menos 8 caracteres que incluyan números y letras, también se exige que la contraseña contenga al menos una letra mayúscula y una letra minúscula. Vea en el siguiente ejemplo el código para realizar esta validación. El siguiente código solo admite contraseñas de entre 8 y 10 caracteres, la contraseña deberá contener cifras y letras y no puede contener caracteres especiales. Vea el ejemplo a continuación:

```
(?!^[0-9]*$)(?!^[a-zA-Z]*$)^([a-zA-Z0-9]{8,10})$
```

Para finalizar, vea una página de Internet que se llama regex101 que permite comprobar las expresiones regulares en múltiples lenguajes de programación como JavaScript, PHP, Python, etc. Con la finalidad de poder saber si la expresión regular que ha creado ejecuta la validación de forma correcta. Vea cómo se muestra este portal en la siguiente imagen:



Como puede ver, en la columna izquierda puede elegir el lenguaje de programación con el que está programando su expresión regular, mientras que en la columna derecha describirá qué funciones tiene las partes de la expresión regular que haya ingresado anteriormente en la barra de arriba. Esta herramienta es muy útil a la hora de crear expresiones regulares complejas y, por lo tanto, debe tenerla muy en cuenta.

Uso de expresiones regulares en HTML5

Si conoce la trayectoria de HTML, debe saber que en su última versión que es HTML5 se han introducido cambios en la manera de crear páginas web, puesto que, se han implementado nuevas y potentes funcionalidades. Una de las nuevas funcionalidades es el atributo pattern, el cual se utiliza para validar elementos input de tipo texto, url, search, email y password.

Este atributo realiza una búsqueda en un campo de texto de una expresión regular o patrón y determina si este cumple el criterio o no. Por consiguiente, la utilidad de este atributo es la de validar datos que hayan sido introducidos por los usuarios sin necesidad de que JavaScript esté activo en el navegador. Vea el siguiente código de ejemplo del uso de pattern en HTML5:

```
<form action="Facebook.php">

<label for="cuenta">Cuenta de Facebook:</label>

<input type="text" pattern="^A-Za-z0-9_{1,10}$" name="cuenta_de_facebook" id="Facebook"/>

<input type="submit" value="Enviar"/>
```

</form>

Si el usuario introduce correctamente los datos, es decir, utiliza caracteres alfanuméricos y guion bajo hasta un máximo de 10 caracteres, el formulario se enviará. Sin embargo, si estos criterios no se cumplen, aparecerá un mensaje de error indicándole al usuario que debe modificar el texto.

Más adelante, volverá a tocar el tema de la validación de datos automática introducida por HTML5 mediante la definición de datos en los campos input.

Ejemplo de validación en comentarios en un blog con jQuery Validate

A continuación, va a ver cómo realizar una validación en un formulario que está pensado para dejar un comentario en una entrada de blog. Los campos que se requerirán serán nombre, email y el comentario a escribir. En el caso del campo para el nombre, la longitud mínima admitida será de 4 caracteres y la longitud máxima serán 30 caracteres. Por otra parte, en el caso del comentario, se establece que la longitud mínima será de 15 caracteres.

En este caso, se utiliza un *plug-in* llamado jQuery Validation. Este *plug-in* es uno de los mejores y más completos *plug-ins* de validación de formularios que existe. La ventaja principal es que es totalmente personalizable, puesto que, da la opción de crear métodos propios además de que cuenta con nuevas actualizaciones constantemente y, también, tiene soporte para varias versiones de jQuery.

Para comenzar, va a ver a continuación el código para definir el formulario del ejemplo que ha dado anteriormente. El código sería el siguiente:

```
<form class="formulario">

<fieldset>

    <legend class="lg">Por favor, déjenos un comentario en su blog</legend>

    <div>

        <label for="name">Nombre</label>

        <input class="required" id="name" title="Su nombre" type="text" name="name"/>

    </div>

    <div>

        <label for="email">Correo electrónico</label>

        <input class="required" id="email" title="Su correo electrónico" type="email" name="email"/>

    </div>

</form>
```

```
<div>

    <label for="comment">Comentario</label>

    <textarea class="required" id="comment" title="Su comentario" type="text" name="comment" rows="6" cols="30"></textarea>

</div>

<div class="submit">

    <input type="button" value="Enviar"/>

</div>

</fieldset>

</form>
```

En este ejemplo, se ha optado por encerrar los campos en un *fieldset*, pero esto no es necesariamente obligatorio. Además, cada campo está encerrado en su div correspondiente, puesto que, esta opción es más elegante que utilizar un espacio *
* para hacer la separación de los mismos. Hay que tener en cuenta que es importante que los parámetros *for* de los *label*, coincidan con los *id* de los campos del formulario y con los *name*.

Lo próximo que va a hacer es crear un código CSS para maquetar, de manera más elegante y atractiva al ojo, su formulario. Se sitúa el formulario en el centro de la página con un fondo azul claro y un borde de 1 píxel de color gris. Vea el código a continuación:

```
.formulario
{
    width: 700px;
    margin: 50px auto;
    border: 1px solid gray;
    background-color: lightblue
}
```

A continuación, se van a poner los campos del formulario con una anchura del 100% y con los *label* en la parte superior. Por otra parte, los mensajes de validación aparecerán en la parte inferior de los campos. Al utilizar el *plug-in* de jQuery, este es el sitio donde aparecen por defecto, pero también

puede personalizarlo para que aparezcan de otra forma. Todo ello, viene muy bien explicado en la documentación del *plug-in*. También, va a definir que cuando un campo del formulario tenga foco, muestre un efecto de color para mejorar, así, la experiencia de usuario. Vea el siguiente código para efectuar esto:

Input [type=text], input [type=email], textarea

{

Padding:10px;

Border:1px solid gray;

Border-radius:2px;

Display-block;

Width:100%;

Font-size:14px;

Outline:none

}

Input:focus, textarea:focus

{

Box-shadow: 0 0 4px #00eecc

}

Al usar el *plug-in* jQuery Validate, se validarán por defecto todos los campos que sean obligatorio, es decir, los campos que haya indicado como *required*. En el caso de que quiera validar un campo como email, como una cifra numérica o quiera definir un número mínimo o máximo de caracteres, debe utilizar el método rules. Para hacer esto, tan solo debe añadir la clase CSS correspondiente en el campo o campos que quiere que cumplan determinada restricción. Por lo tanto, para definir en método rules, va a utilizar una serie de reglas de validación que pasará por parámetro mediante un objeto que asignará las reglas a cada campo del formulario. Para ello, verá como el ejemplo el código siguiente:

```
$('.formulario').validate({
```

```
    Rules:{
```

```
campo1: {  
    regla1 : valor1,  
    regla2 : valor2,  
    [todas las reglas que queramos definir más]  
},  
  
campo2: {  
    regla1 : valor1,  
    regla2 : valor2,  
    [todas las reglas que queramos definir más]  
},  
  
[todos los campos que queramos incluir]  
  
campoN :{  
    regla1 : valor1,  
    regla2 : valor2,  
    [todas las reglas que queramos definir más]  
}  
});
```

Por ejemplo, si necesita validar un campo tipo email que es obligatorio, tan solo debe definir la siguiente regla. Véala en el siguiente código:

```
$('.formulario').validate({  
  
campoEmail: {  
    required: true,  
    email: true  
}
```

A continuación, verá una lista completa de reglas de validación que puede utilizar con jQuery Validate:

Regla de validación	Valores admitidos
Required	True/false
Email	True/false
Digits	True/false
Creditcard	True/false
Min	Numérico
Max	Numérico
Minlength	Numérico
Maxlength	Numérico
Rangelength	Ejemplo. Rangelength [1,9]
url	True/false
Date	True/false
dateISO (Fecha en formato internacional)	True/false
dateDE (Fecha en formato alemán)	True/false
phoneUS (Teléfono en formato USA)	True/false
Number	True/false
numberDE (número en formato alemán)	True/false
Accept	Ejemplo. "xls csv" (tipos de ficheros permitidos)
equalTo (igual a otra cadena de texto)	Ejemplo. "#password"
Remote (opciones de cadena en remoto)	Opciones \$.ajax

Volviendo al ejemplo del comentario de blog, para validar las entradas de su formulario de comentario de blog, necesita las reglas de validación *required* para todos los campos, email para el campo correspondiente, minlength y maxlength para las restricciones en el número de caracteres tanto en el campo nombre como en el campo del comentario. Además de establecer las reglas de validación, debe definir los mensajes de error que se mostrarán en el caso de que no se cumplan los criterios establecidos para cada campo. Esto lo se hará pasando como parámetro un objeto llamado messages que relaciona cada campo del formulario y cada criterio con un mensaje de texto determinado que, al no cumplirse los criterios, se mostrará. Vea a continuación cómo programaría este objeto:

Messages:{

Campo1: {

Regla1: "mensaje de error 1",

Regla2 : "mensaje de error 2",

[Todos los mensajes de error que necesitemos programar]

},

Campo2: {

Regla1: "mensaje de error 3",

Regla2 : "mensaje de error 4",

[Todos los mensajes de error que necesitemos programar]

},

[Todos los campos que queramos programar]

campoN: {

regla1 : "mensaje de error M-1"

regla2 : "mensaje de error M",

[...]

}

A continuación, se verá cómo quedaría el código JavaScript que define las reglas de validación, junto con el código que definen los mensajes de error correspondientes. Véalo en el siguiente ejemplo:

```
<script src="http://ajax.googleapis.com/ajax/libs/jquery/1.10.2/jquery.min.js"></script>

<script src="http://ajax.aspnetcdn.com/ajax/jquery.validate/1.11.1/jquery.validate.js"></script>

<script>

$(function(){

  $('.formulario').validate({

    rules:{

      email:{

        required: true,

        email: true

      }

    }

  });

});
```

```
name:{  
    required: true,  
    minlength: 4,  
    maxlength: 30  
},  
  
comment:{  
    required: true,  
    minlength: 15  
}  
,  
  
messages: {  
    email:{  
        required:"Es necesario introducir un email",  
        email: "El email introducido no es válido"  
    },  
  
    name: {  
        required: "Es necesario introducir un nombre",  
        minlength: "Longitud mínima: 4 caracteres",  
        maxlength: "Longitud máxima: 30 caracteres"  
    },  
  
    comment: {  
        required:"No ha escrito ningún comentario",  
        minlength: "Longitud mínima: 15 caracteres"  
    }  
}
```

```
    }  
}  
});  
});  
  
</script>
```

En el ejemplo que ha visto, las dos primeras líneas de código son necesarias para que todo el resto del código funcione, ya que, la primera línea llama a la librería jQuery de JavaScript que está alojada en el repositorio de Google APIs, por otro lado, la segunda línea llama *al plug-in* JQuery Validate que se encuentra alojado en aspnetcdn.com. Sin embargo, además de este método de uso de las librerías JavaScript, también puede descargarlas, almacenarlas en su servidor o en su ordenador de manera local y hacer referencia a esta copia desde el código que está programando.

A continuación, verá el código completo de este ejemplo, en él se inserta CSS adicional, como verá ahora. Se utilizan las librerías JavaScript necesarias en remoto, así su código podrá funcionar sin necesidad de descargar nada adicional. Vea el ejemplo a continuación:

```
<!DOCTYPE html>  
  
<html lang="es">  
  
  <head>  
    <meta charset="utf-8">  
  
    <script src="http://ajax.googleapis.com/ajax/libs/jquery/1.10.2/jquery.min.js"></script>  
  
    <script src="http://ajax.aspnetcdn.com/ajax/jquery.validate/1.11.1/jquery.validate.js"></script>  
  
    <script>  
      $(function(){  
        $('#formulario').validate({  
          rules:{  
            email:{  
              required: true,  
              email: true  
            }  
          }  
        })  
      })  
    </script>  
  </head>  
  
  <body>  
    <form id="formulario">  
      <input type="text" name="email" value="user@example.com" />  
      <input type="submit" value="Enviar" />  
    </form>  
  </body>  
</html>
```

```
    },  
  
    name:{  
  
        required: true,  
  
        minlength: 4,  
  
        maxlength: 30  
    },  
  
    comment:{  
  
        required: true,  
  
        minlenth: 15  
    },  
  
    messages: {  
  
        email:{  
  
            required:"Es necesario introducir un email",  
  
            email: "El email introducido no es válido"  
        },  
  
        name: {  
  
            required: "Es necesario introducir un nombre",  
  
            minlength: "Longitud mínima: 4 caracteres",  
  
            maxlength: "Longitud máxima: 30 caracteres"  
        },  
  
        comment: {  
  
            required:"No ha escrito ningún comentario",  
        }  
    }  
}
```

```
minlength: "Longitud mínima: 15 caracteres"

}

});

});

</script>

<style>

body {

font-family: Arial;

background-color: lightgray

}

div{

clear: both;

padding-bottom:25px;

position:relative

}

.formulario {

width: 700px;

margin: 50px auto;

border: 1px solid gray;

background-color: lightblue;

}

</style>
```

```
</head>

<body>

</body>

</html>
```

Ejemplo de validación en comentarios en un blog con Verify.js

En este apartado va a ver cómo sería la validación de un formulario utilizando Verify.js. Primero, debe modificar la línea que incluye el *plug-in*. Como Verify.js también utiliza la librería de JQuery para funcionar, tan solo debe modificar la segunda línea que era la que contenía el *plug-in* JQuery Validate. Para ello, vea el siguiente ejemplo donde se usa una variante de Verify.js que se llama notify.js que se encargará de mostrar los mensajes de validación. Vea en el siguiente código:

```
<script src="//raw.github.com/jpillora/verifyjs/gh-pages/dist/verify.notify.min.js"></script>
```

Vea a continuación un sencillo ejemplo de validación utilizando esta herramienta:

```
<form>

<input name="edad" id="edad" data-validate="number">

<input name="puntos" id="puntos" data-validate="number">

<input type="submit" class="submit">

</form>
```

En comparación con la herramienta que se utilizaron anteriormente, JQuery Validate, la herramienta Verify.js funciona de manera diferente, ya que, hace una validación asíncrona. Para poder realizar la validación asíncrona, esta herramienta utiliza un sistema parecido al de Parsley, para ello, establece una serie de atributos de *data-attributes* para cada campo del formulario y, de este modo, indica cómo y qué se debe validar. De igual forma, también se pueden agrupar validaciones.

Esta librería inicia cualquier elemento del formulario que contenga el atributo *data-validate* de manera automática. Por otro lado, si sirve la configuración que viene dada por defecto, no tendrá la necesidad de programar nada de manera adicional. En la siguiente tabla puede ver un resumen de las reglas de validación implementadas por defecto en Verify.js. Véala a continuación:

Regla de validación	Valores admitidos	Validación
Required	Ninguno	Elemento obligatorio
Email	Ninguno	Validación de email
url	Ninguno	URL válida
Number	Ninguno	Solo dígitos
numberSpace	Ninguno	Solo dígitos con espacios en blanco
alphaNumeric	Ninguno	Solo dígitos y letras
Currency	Ninguno	Valida valores de moneda
Decimal	Decimales (2 por defecto)	Valida número con decimales. Redondea en función del parámetro
Regex o pattern	Expresión regular, mensaje de error	Valida la expresión regular especificada
Min	Valor numérico	Número mínimo de caracteres
Max	Valor numérico	Número máximo de caracteres
Size	Valor numérico	Número exacto de caracteres
Size	Mínimo, máximo	El número de caracteres debe estar entre un rango especificado
minVal	Valor numérico	El valor introducido en el campo debe ser mayor o igual al valor numérico especificado
maxVal	Valor numérico	El valor introducido en el campo debe ser menor o igual al valor numérico
rangeVal	Mínimo, máximo	El valor debe estar entre un rango de valores especificado
Agreement	Ninguno	Aceptación de unas condiciones determinadas mediante una checkbox
minAge	Valor numérico	Edad mínima

En resumen, para validar un formulario de comentarios para un blog, tendría que cambiar un poco el código HTML que vio anteriormente. Vea el ejemplo a continuación:

```
<form class="formulario">
<fieldset>

<legend class="lg">Por favor, déjenos un comentario en su blog</legend>

<div>
<label for="name">Nombre</label>
```

```
<input data-validate="required, size(4,30)" id="name" title="Su nombre" type="text"  
name="name"/>  
  
</div>  
  
<div>  
  
<label for="email">Correo electrónico</label>  
  
<input data-validate="required,email" id=email" title="Su correo electrónico" type="email"  
name="email"/>  
  
</div>  
  
<div>  
  
<label for="comment">Comentario</label>  
  
<textarea data-validate="required,min(15)" id="comment" title="Su comentario" type="text"  
name="comment" rows="6" cols="30"/></textarea>  
  
</div>  
  
<div class="submit">  
  
<input type="button" value="Enviar"/>  
  
</div>  
  
</fieldset>  
  
</form>
```

Ejemplo de validación en comentarios en un blog con Parsley.js

Para utilizar la validación con Parsley.js, debe descargar la librería de su propia web. Una vez haya descargado la librería, debe introducir la siguiente línea de código:

```
<script src="parsley.min.js"></script>
```

Además de la línea de código de Parsley.js, debe mantener la línea de JQuery para que Parsley pueda funcionar, ya que, este necesita de la librería de JQuery en su versión 1.8 o superior.

Como ya ha visto, Parsley.js utiliza una API que se utiliza a través del DOM. Para hacer posible esta función, se utiliza un prefijo por defecto mediante el que añadir funcionalidades a las entradas del formulario, e indicar cómo quiere que sean validadas estas entradas. El prefijo que se usa para

realizar todo esto es el data-parsley-. Por lo tanto, si tiene una propiedad llamada email se escribe data-parsley-email para poder acceder a ella y realizar funciones con ella.

A continuación, verá las reglas que hay que tener en cuenta para utilizar la API del DOM de Parsley.js:

Primeramente, esta API utiliza camelization, al igual que JQuery &.data(), este término se podría traducir como camellización, término que hace una analogía entre la alternancia de mayúsculas y minúsculas con las jorobas de un camello. Por ejemplo, si escribe nombre de usuario utilizando camelization quedaría como nombreDeUsuario. Por lo tanto, si en Parsley.js tiene una propiedad llamada data-parsley-nombre-usuario, en el código se correspondería con una propiedad denominada nombreUsuario.

Por otra parte, cada propiedad que está establecida en el DOM de Parsley.js está guardada en this.options para cada instancia.

Por último, puede cambiar el namespace directamente en el DOM mediante la siguiente instrucción data-parsley-namespace="campos-validacion-". Por lo tanto, si toma como referencia el ejemplo anterior del campo nombre de usuario, la sentencia quedaría como campos-validacion-nombre-usuario. Escribir los atributos de esta manera permite reutilizar el código e impide que haya colisiones entre diferentes formularios y validaciones. Por ejemplo, si tiene un formulario de comentarios de blog y otro de contacto que, a su vez, tienen campos comunes como nombre, email, etc. si no cambia el namespace podrían colisionar entre sí, ya que, el nombre del campo sería el mismo.

En conclusión, va a ver cómo quedaría la estructura de la instalación y configuración de Parsley.js a continuación:

```
<script src="jquery.js"></script>

<script src="parsley.min.js"></script>

<form id="formulario">

[...]

</form>

<script type="text/javascript">$('#formulario').parsley();</script>
```

Debido a que este script proporciona los mensajes de validación por defecto en inglés, la comunidad ha hecho posible que se pueda cambiar el idioma por defecto. Por lo tanto, si quiere que aparezcan los mensajes de validación en español, debe cambiar un poco la estructura de la instalación. Vea el ejemplo a continuación:

```
<script src="jquery.js"></script>
```

```
<script src="i18n/fr.js"></script>  
  
<script src="i18n/es.js"></script>  
  
<script src="parsley.min.js"></script>  
  
<script type="text/javascript">  
  
Window.ParsleyValidator.setLocale('es');  
  
</script>
```

Una de las facilidades que ofrece Parsley.js, precisamente es lo fácil que es cambiarlo de idioma. En este caso ha instalado dos idiomas francés y español, pero solo ha activado los mensajes en español. Para finalizar, va a ver lo que contiene el fichero es.js. Este fichero contiene la definición completa de los mensajes de error de las funciones de validación incluidas por defecto y, además, son ampliables. A continuación, verá su contenido:

```
//ParsleyConfig definition if not already set  
  
Window.ParsleyConfig = window.ParsleyConfig || {};  
  
Window.ParsleyConfig.i18n = window.ParsleyConfig.i18n || {};  
  
Window.ParsleyConfig.i18n.es = $.extend(window.ParsleyConfig.i18n.es || {}, {  
  
    Default message: "Este valor parece ser inválido.",  
  
    Type: {  
  
        Email: "Este valor debe ser un correo válido.",  
  
        url: "Este valor debe ser una URL válida.",  
  
        number: "Este valor debe ser un número válido.",  
  
        integer: "Este valor debe ser un número entero válido.",  
  
        digits: "Este valor debe ser un dígito válido.",  
  
        alphanum: "Este valor debe ser alfanumérico."  
  
    },  
  
    Notblank: "Este valor no debe estar en blanco.",
```

Required: "Este valor es requerido.",

Pattern: "Este valor es incorrecto.",

Min: "Este valor no debe ser menor que %s.",

Max: "Este valor no debe ser mayor que %s.",

Range: "Este valor debe estar entre %s y %s.",

Minlength: "Este valor es muy corto. La longitud mínima es de %s caracteres.",

Maxlength: "Este valor es muy largo. La longitud máxima es de %s caracteres.",

Length: "La longitud de este valor debe estar entre %s y %s caracteres.",

Mincheck: "Debe seleccionar al menos %s opciones.",

Maxcheck: "Debe seleccionar %s opciones o menos.",

Rangecheck: "Debe seleccionar entre %s y %s opciones.",

Equalto: "Este valor debe ser idéntico."

});

//If file is loaded after PArsley main file, auto-load locale

If ('undefined' !== typeof window.ParsleyValidator)

Window.ParsleyValidator.addCatalog ('es', window.ParsleyConfig.i18n.es,true);

Cuando haya finalizado de instalar y configurar el componente, ya solo queda modificar el código HTML para que Parsley.js sepa qué entradas son las que espera que introduzca el usuario en cada campo del formulario. Tomando el ejemplo anterior, debe modificar el HTML de la siguiente manera:

```
<form class="formulario" data-parsley-validate>
```

```
  <fieldset>
```

```
    <legend class="lg">Por favor, déjenos un comentario en su blog</legend>
```

```
    <div>
```

```
      <label for="name">Nombre</label>
```

```
<input data-parsley-length="[4,30]" data-parsley-trigger="change" required id="name" title="Su nombre" type="text" name="name"/>

</div>

<div>

<label for="email">Correo electrónico</label>

<input data-parsley-type="email" data-parsley-trigger="change" required id="email" title="Su correo electrónico" type="text" name="email"/>

</div>

<div>

<label for="comment">Comentario</label>

<textarea data-parsley-minlength="15" data-parsley-trigger="change" required id="comment" title="Su comentario" name="comment" rows="6" cols="30"/></textarea>

</div>

<div class="submit">

<input type="button" value="Enviar"/>

</div>

</fieldset>

</form>
```

La nueva instrucción que ha añadido `data-parsley-trigger="change"` sirve para controlar el comportamiento que quiere que tenga Parsley.js al validar su formulario. En su caso, lo ha programado para que se realice la validación cada vez que se cambia el contenido del campo, aunque este no haya perdido el foco. Esta práctica no es la más óptima desde el punto de vista del usuario, puesto que, van a estar mostrándose mensajes de error desde el principio. Por lo tanto, si quiere evitar este comportamiento tan intrusivo con el usuario, debe cambiar la instrucción mencionada anteriormente por `data-parsley-trigger="focusin focusout"`. Esta nueva expresión hará que solo se valide el contenido del campo cuando se introduzca algún contenido en él y posteriormente se cambie de foco.

A continuación, verá una lista de validaciones disponibles que es un poco más extensa de lo normal. Debe tener en cuenta que pueda añadir nuevas validaciones usando la instrucción `window.ParsleyConfig` para definir una nueva validación y la instrucción

window.ParsleyValidator.addValidator para registrarla. Vea a continuación una tabla con las validaciones incluidas por defecto:

Nombre	Etiqueta DOM API	Descripción
Required	Data-parsley-required	Campo obligatorio
Email	Data-parsley-type="email"	Validación de email
Number	Data-parsley-type="number"	Validación de tipo numérico
Integer	Data-parsley-type="integer"	Validación de número entero
Digits	Data-parsley-type="digits"	Validación solo de cifras
Alphanum	Data-parsley-type="alphanum"	Validación de cadena alfanumérica
url	Type="url"	Validación de url válida
Minlength	Data-parsley-minlength="8"	Mínimo de caracteres permitidos para la cadena
Maxlength	Data-parsley-maxlength="50"	Máximo de caracteres permitidos para la cadena
Length	Data-parsley-length="[8,20]"	Número caracteres permitidos comprendido en el intervalo
Min	Data-parsley-min="3"	Número dado debe ser mayor o igual a un valor
Max	Data-parsley.max="11"	Número dado debe ser menor o igual a un valor
Range	Data-parsley-range="[8,20]"	Número dado debe estar dentro del rango
Pattern	Data-parsley-pattern="\d+"	El valor del campo debe cumplir un patrón. Recuerde RegExp.
MinCheck	Data-parsley-mincheck="5"	Valida que se seleccionen un número de opciones mínimas para un checkbox
MaxCheck	Data-parsley-maxcheck="2"	Valida que se seleccionen un número máximo de opciones en un checkbox
Check	Data-parsley-check="[2,6]"	Valida que se seleccionen un número de opciones de un checkbox dentro de un rango.
Equalto	Data-parsley-equalto="#anotherfield"	Valida que el valor del campo sea igual al de otro campo especificado

Ejemplo de validación en comentarios en un blog con Mootools FormValidator

Para realizar la validación de su código con Mootools, lo primero que debe hacer es cambiar la línea de código donde se incluía JQuery, y sustituirla por la de Mootools. En este ejemplo y para no tener que descargar nada, va a incluir Mootools haciendo una llamada a la versión que está alojada en Google Hosted Libraries. Para ello, va a añadir la siguiente línea de código:

```
<script src="//ajax.googleapis.com/Ajax/libs/mootools/1.5.0/mootools-yui-compressed.js">
```

Como ha visto anteriormente, la herramienta Mootool no necesita de ningún *plug-in* para funcionar, puesto que ya viene con el componente FormValidator que es el que va a utilizar para añadir las validaciones a su formulario de ejemplo.

Para validar los diferentes campos con Mootools, debe configurar los campos enumerando una serie de clases de forma parecida a como vio en JQuery Validate. Por lo tanto, para escribir un email debe insertar la siguiente línea de código:

```
<input class="required validate-email campoEmail" type="text" name="profile[email]"/>
```

En esta línea, ha especificado que el campo es obligatorio, para ello ha usado required. También, ha especificado que lo que va a introducir en ese campo es de tipo correo electrónico, por ello ha escrito validate-email y, por último, ha especificado que una validación personalizada como campoEmail, nombre que escoge el usuario.

Por otra parte, si quiere definir una longitud mínima en la entrada de un campo, se utiliza la siguiente instrucción:

```
<input type="text" class="minLength:8" name="profile[firstname]"/>
```

Anteriormente, ha añadido una línea de código donde ha introducido una validación personalizada. Para hacer que la validación personalizada funcione, debe definirla utilizando FormValidator.add. Lo que va a hacer con esto es comprobar utilizando una validación asíncrona, si el correo electrónico que introduzca el usuario ha sido utilizado previamente o no. Para realizar esto, va a llamar a una rutina llamada fValidator.php que proporcionará una respuesta. Vea el código a continuación:

```
FormValidator.add('campoEmail',  
{  
    errorMsg:'La cuenta de correo ya ha sido registrada en el sistema',  
    test: function (element, props)  
    {  
        if (element.value.length>0)
```

```
{  
    var req=newRequest({  
        url:'/fValidator.php',  
        async: true  
    }).send("mail="+element.value);  
  
    return (req.response.text !='1');  
}  
  
return true;  
};  
});
```

Por otro lado, los mensajes de error se pueden modificar fácilmente utilizando código CSS, ya que, Mootools contiene una serie de clases por defecto donde las propiedades son fácilmente modificables. Vea el siguiente ejemplo:

```
/*Estilo que se aplica a los campos cuando hayan pasado la validación*/  
  
.validation-passed {  
    background-color:#74ed74 !important;  
  
    //Si no utiliza !important, no funcionaría  
}  
  
/*Estilo que se aplica a los mensajes de error a validar */  
  
.validation-advice{  
    background-color: yellow;  
  
    margin: 10px;  
  
    padding: 5px;  
}
```

```
/*Estilo que se aplica a los campos cuando no pasan la validación */  
  
.validation-failed {  
  
    background-color: red;  
  
    color: white  
  
}
```

A continuación, va a modificar su código HTML para que funcione con el componente FormValidator. Véalo a continuación:

```
<div class="submit">  
  
<input type="button" value="Enviar"/>  
  
</div>  
  
</fieldset>  
  
</form>
```

Para finalizar, necesita añadir el código JavaScript para que pueda ejecutarse la validación cuando se haya cargado el DOM. Para ello, se observa cómo se hace el siguiente código:

```
window.addEvent('domready',function()  
  
{  
  
    miValidacion = new Form.Validator.Inline($('#formulario'),  
  
    {  
  
        stopOnFailure: true,  
  
        useTitles: true,  
  
        errorPrefix: "Error",  
  
        onFormValidate: function(passed, form, event)  
  
        {  
  
            if (passed){
```

```
    form.submit();  
}  
}  
});  
});
```

Para terminar, va a ver una tabla donde puede ver las validaciones disponibles de FormValidator. Véalo a continuación:

Regla de validación	Descripción
Required	Campo obligatorio
Validate-email	Validación de email
Validate-integer	Validación de entero
Validate-digits	Validación solo de dígitos
Validate-numeric	Validación de números reales
Validate-date	Validación de fecha
Minlength	El campo debe tener un mínimo de caracteres
Maxlength	El campo debe tener un máximo de caracteres
Validate-url	Validación de url
Validate-alpha	Validación de campo alfabético
Validate-alphanum	Validación de campo alfanumérico
dateDE True/False	Fecha formato alemán
phoneUS	Teléfono de USA
Number	Formato numérico
numberDE	Formato numérico alemán
Accept	Tipos de ficheros permitidos
equalTo	Igualar a otra cadena de texto
Remote	Opciones de cadena en remoto, por ejemplo: ajax

1.1.7. Ejecución del código de validación

La manera clásica de validar un formulario es ejecutar la validación una vez el usuario pulsa el botón enviar. Pero realmente esto no es práctico y, ni mucho menos, eficiente. Es por ello que lo que se debe hacer es que antes de proceder al envío del formulario se realice una validación en el lado del cliente.

Para ello, se utilizará JavaScript, el cual alertará al usuario de los posibles campos erróneos que haya introducido. Así, al enviar el usuario el formulario sabrá que los campos son correctos y, de esta forma, le será más fácil al usuario modificar esos campos erróneos.

Validación con JavaScript de un campo cuando pierde el foco

Si quiere validar cada campo del formulario cuando estos pierden el foco y sin necesidad de que el usuario clique el botón “Enviar”, se utiliza JavaScript. En este tipo de validación, cuando el usuario cambie el foco del campo, se mostrará un mensaje junto a ese campo indicando si lo que ha introducido el usuario es correcto o no.

Para realizar este tipo de validación, se utiliza una librería como JQuery para poder capturar los eventos necesarios para que los campos del formulario se comporten de manera que cuando se realice algún cambio en ellos, se ejecute la función de validación.

Antes de comenzar a ver ejemplos con JQuery, verá ejemplos con JavaScript puro, es decir, sin usar ninguna librería añadida. Para realizar este ejemplo, usa el método getElementById. Este método pertenece al objeto document y puede acceder a él a través del DOM. Por otra parte, con código CSS va a añadir una clase llamada .mensaje y le pone la propiedad display = none, esta propiedad hará que los mensajes dirigidos al usuario no se muestren hasta que sea necesario. A continuación, vea cómo quedaría esta clase:

```
.mensaje {display:none}
```

A continuación, va a asignar cada elemento del formulario a una variable como se muestra en el siguiente ejemplo:

```
Var elementoFormulario=document.getElementById("idElemento");
```

El método getElementById crea un enlace entre el id del objeto y el id del elemento. Por lo tanto, debe haber especificado el mismo id en su código HTML. Vea el siguiente ejemplo.

```
<div id="capa1">Texto</div>
```

En el caso de que quiera cambiar el color del texto, debe utilizar la línea de JavaScript que se muestra a continuación:

```
Document.getElementById('capa1').style.color="blue";
```

Además, tomando el ejemplo anterior como base, puede lanzar una función que muestre un mensaje cuando cambia el contenido de un campo de la misma manera. En este ejemplo, se usa directamente el elementoFormulario, pues anteriormente ha hecho que la variable elementoFormulario apunte hacia la id del elemento que quiera controlar. Vea a continuación el código de ejemplo:

```
elementoFormulario.onchange = function() {  
  
if (this.value == "") eMensaje.style.display = "block";}
```

A continuación, verá el código completo del ejemplo anterior. En este código se comprobará que el nombre de usuario tenga al menos 3 caracteres y que la edad esté expresada obligatoriamente en cifras. Vea el código HTML siguiente:

```
<form>

<div class="fLeft">

<label for="nombre">Nombre: </label>

<input type="text" id="nombreUsuario" name="nombre">

</div>

<div class="mensajefLeft" id="nombreErroneo">El nombre debe contener al menos 3 caracteres</div>

<div class="mensajefLeft" id="nombreValido">El nombre introducido es válido</div>

<div class="clear"></div>

<div class="fLeft">

<label for="edad">Edad: </label>

<input type="text" id="edadUsuario" name="edad">

</div>

<div class="mensajefLeft" id="edadErronea">La edad se debe expresar en cifras</div>

<div class="mensajefLeft" id="edadValida">Edad correcta</div>

<div class="clear"></div>

<input type="button" value="Enviar"></input>

</form>
```

Lo próximo que hará es programar un código CSS para maquetar lo que ha programado en HTML y darle un diseño más atractivo a la vista. En el CSS va a definir un identificador para cada mensaje. Para ello, va a definir #nombreErroneo y #edadErronea y le pondrá el texto de color rojo, ya que, son mensajes de error. Por otra parte, para #nombreValido y #edadValida pondrá el texto en color verde. También, se utiliza los identificadores #nombreUsuario y #edadUsuario para maquetar estos elementos en CSS.

Vea el ejemplo a continuación:

```
<style>

.mensaje {display: none; margin-left: 10px}

.fLeft {float: left}

.clear {clear: both}

#nombreErroneo, #edadErronea {color: red}

#nombreValido, #edadValida {color: green}

</style>
```

A continuación, debe programar el código JavaScript y lo primero que debe hacer es definir las funciones que manejarán los distintos eventos. Por lo tanto, si quiere que, al cambiar el contenido de los campos de texto, se compruebe que el propio contenido es válido, debe utilizar el texto que se expone a continuación:

```
comprobarNombre = function()

{

    if (this.value.length<3)

    {

        //Aquí se mostrará el mensaje de error correspondiente

        //y se ocultará el mensaje de conformidad

        nombreErroneo.style.display="block";

        nombreValido.style.display="none";

        //Cambiemos el foco al campo de texto

        this.focus();

    }

    Else

    {
```

```
//Aquí se muestra el mensaje de conformidad

//Y se oculta el mensaje de error

nombreErroneo.style.display="none";

nombreValido.style.display="block";

}

};

comprobarEdad = function()

{

valor = rhis.value.valueOf();

//Comprobar si el valor es numérico

if (isNaN(valor) | |this.value.length==0)

{

//Si el valor introducido no es numérico o está vacío

//Se muestra el mensaje de error correspondiente

//Y se oculta el mensaje de conformidad

edadErronea.style.display="block";

edadValida.style.display="none";

this.value="";

this.focus();

}

Else

{

//Aquí se muestra el mensaje de conformidad
```

```
//Y se oculta el mensaje de error  
  
edadErronea.style.display="none";  
  
edadValida.style.display="block";  
  
}  
  
};
```

Llegados a esta parte, se van a crear diversas variables que apunten a los elementos que quiere controlar mediante el método getElementById(). Para ello, vea el siguiente ejemplo:

```
var inNombre = document.getElementById("nombreUsuario");  
  
var inEdad = document.getElementById("edadUsuario");  
  
var nombreValido = document.getElementById("nombreValido");  
  
var edadValida = document.getElementById("edadValida");  
  
var nombreErroneo = document.getElementById("nombreErroneo");  
  
var edadErronea = document.getElementById("edadErronea");
```

Por último, lo que hará es asignar los eventos onchange a las funciones que manejarán a las variables que ha creado anteriormente. Para ello, utiliza el siguiente código:

```
inNombre.onchange = comprobarNombre;  
  
inEdad.onchange = comprobarEdad;
```

Validación de un campo al mismo tiempo en que son introducidos los datos

También, se pueden restringir el conjunto de caracteres válidos de entrada para un campo determinado, de esta forma solo se mostrarán los caracteres permitidos. Por ejemplo, imagine un campo donde haya que introducir la edad en dígitos, en este campo podría programar una validación que solo muestre los datos introducidos cuando el usuario teclee números del 0 al 9. Si el usuario teclea otra cosa que no sean números, no aparecerá ningún dato en el campo. Para realizar este tipo de validación se fijará en el ejemplo siguiente. En este ejemplo, el campo de texto está desactivado para las entradas que no sean numéricas. La función que aparece, comprueba que la tecla pulsada dentro del input esté dentro del rango de teclas permitidas. Véalo a continuación:

```
<script>  
  
function soloNumeros (event)
```

```
{  
    var key = window.event ? event.keyCode : event.which;  
  
    if (event.keyCode== 8 || event.keyCode== 46 || event.keyCode == 37 || event.keyCode == 39)  
  
    {  
        return true;  
    }  
  
    else if (key < 48 || key > 57){  
  
        return false;  
    }  
  
    else return true;  
};  
  
</script>  
  
//En el input se llama a soloNumeros (event)  
  
//cuando la tecla aún no ha sido liberada (onkeydown)  
  
<label for="telefono">Teléfono:</label>  
  
<input type="input" id="telefono" maxlength="11" onkeydown="soloNumeros(event)"/>
```

Validar campos de formulario definiendo los tipos de datos en HTML5

Anteriormente, ha visto un ejemplo en el que no se necesitaba la validación de expresiones regulares mediante JavaScript, ya que, el HTML5 tiene nuevas características que permiten hacerlo. Pues, también se debe saber que la nueva versión de HTML5 permite ejecutar la validación automática de tipos de datos tan solo con definirlos como tal. Vea el siguiente ejemplo:

```
<form action="blog.php">  
  
<label for="correoElectronico">Email: </label>  
  
<input type="email" name="correoElectronico" id="correoElectronico"/>  
  
<input type="submit" value="Enviar"/>
```

</form>

En este código que acaba de ver, se define el recuadro de entrada de datos como <input type="email"/>, por lo tanto, si usa navegadores que sean compatibles con esta característica se mostrará un mensaje de error de validación de forma automática avisando del error si fuera el caso.

Vea, un ejemplo análogo donde ocurre lo mismo en un campo donde se le pide al usuario su edad:

```
<form action="blog.php">

<label for="edad">Edad:</label>

<input type="number" name="edad" id="edad"/>

<input type="submit" value="Enviar"/>

</form>
```

Este tipo de validación será lo común que encuentre en un futuro donde solo será necesario implementar JavaScript para validaciones más complejas, pero en la actualidad debe tener en cuenta que hay muchísimos navegadores que no son compatibles con este tipo de validación.

1.2. Verificar formularios

Como ya se ha mencionado anteriormente, validar no es lo mismo que verificar, por lo que, no es lo mismo validar una fecha que verificarla. Por ejemplo, si tiene que programar una página web de vuelos o de reservas de casas vacacionales, debe comprobar las fechas de llegadas y las fechas de salida y estas deberán cumplir una serie de requisitos. Estos requisitos además de ser que las fechas sean correctas, es que la fecha de salida sea anterior a la fecha de llegada. Y, además si en la web programa que la estancia mínima sean 3 noches, se tendrá que comprobar que las fechas de salida y de entrada cumplen también esta condición.

Un ejemplo muy común de verificación de datos es cuando un usuario quiere acceder al perfil que tiene creado en una web y esta le pide el nombre de usuario y la contraseña. Teniendo este ejemplo en cuenta, por un lado, en el lado del cliente se validará si el nombre de usuario cumple los requisitos en cuanto su forma y, por el otro, en la parte del servidor se verificará si este nombre de usuario existe y si coincide con la contraseña introducida.

No siempre la validación va antes que la verificación de los datos. Un ejemplo para entender esto es, por ejemplo, cuando tiene una web en la que para ver cierto contenido donde haya formularios debe estar registrado primero, habrá que pasar la verificación del nombre y contraseña para acceder al perfil y, después, comenzará la validación de los campos de formulario, ya que, al contrario, no tendría sentido, pues los usuarios que aún no estén registrados no serán capaces de ver los formularios.

Por otro lado, para verificar el acceso de un usuario ya registrado, se deberá ejecutar un *script* en el lado del servidor capaz de acceder a la tabla de usuarios registrado, que se encontrará en la base de datos. Una vez que el *script* haya encontrado los datos, devolverá una respuesta y la página web se recargará para recibirla. Aunque, en la actualidad gracias al desarrollo tecnológico que tiene, es posible hacer el proceso de verificación de datos en el lado del servidor sin recargar la página. Esto puede verlo en uno de los ejemplos que se vieron anteriormente donde se verificaban los campos según el usuario iba introduciendo los datos. Y en muchas páginas web que piden crear un usuario pueda observar que al introducir un nombre de usuario va a dar una respuesta sobre si este existe o no.

Para poder implementar este tipo de comportamiento en su web y evitar, así, la recarga de la misma, debe emplear AJAX, que es un tipo de JavaScript asíncrono y XML. De esta manera puede mantener una comunicación con el servidor y, por lo tanto, tendrá también acceso a la base de datos, todo ello en segundo plano. Así es como puede mejorar la velocidad de la página y la experiencia de usuario de la misma.

1.2.1. Identificación de datos

La identificación de datos es primordial para saber qué datos necesitan ser verificados después de haber pasado una validación. La base del proceso de identificación de datos a verificar dependerá de las necesidades de cada sistema. En el caso del acceso a una web mediante usuario y contraseña es evidente la identificación de datos a verificar. En este caso será imprescindible verificar la existencia o no del nombre de usuario y la equivalencia de la contraseña. Otro ejemplo donde la identificación también es evidente es en el caso donde haya que introducir una tarjeta o un número de cuenta corriente. Por ejemplo, si en una tienda online se necesita realizar un reembolso a un cliente, se deberá siempre reembolsar el dinero a través de los datos que aportó el cliente en el momento de su compra. Para verificar los datos de una tarjeta de crédito, normalmente, se le pide al usuario el código CVV que aparece en la parte de atrás de todas las tarjetas de crédito, y también se le pide la fecha de caducidad de la misma. Esto, en teoría, garantiza que el cliente tiene la tarjeta físicamente en el momento de la compra y que no solo tiene el número de tarjeta, ni esta ha sido duplicada fraudulentamente.

En otros casos no se hace tan fácil la tarea de identificar los datos que se deben verificar, ya que, esto dependerá de una decisión totalmente subjetiva. Un ejemplo de estos casos es, por ejemplo, la suscripción a un boletín informativo donde, aunque no sea un proceso de riesgo y no sea necesario pedir datos de confirmación, siempre se deberá garantizar el cumplimiento de la ley de protección de datos y se deberá permitir al usuario darse de baja cuando quiera. Además, si no se verifican los datos en el proceso de registro en un boletín informativo, también puede ocurrir que un usuario introduzca en la suscripción los datos de un tercero sin su consentimiento. Siguiendo con el caso de los boletines informativos y de la tienda online, el cliente realmente está interesado en dar su email verdadero para poder tener el seguimiento de su pedido o para recibir ofertas y descuentos, pero es cierto que no se puede fiar al cien por cien de que el cliente haya dado un correo válido, ya que, podría haber cometido un error al escribirlo. Por ello, lo que debe hacer para verificar el correo electrónico es enviar un enlace a la dirección de correo que ha aportado el cliente, si el cliente pincha en el enlace puede verificar que el email es correcto si no, debe descartar ese email.

En cambio, hay otros tipos de datos que son más difíciles de verificar, por ejemplo, la edad, ya que, muchas personas son reacias a dar este tipo de dato y lo omiten o dan una edad que no es real. Hay tipos de webs en los que este dato es un dato importante de verificar, ya que, puede quese vendas productos aptos solo para mayores de 18 años por poner un ejemplo, podría tener una tienda online de artículos para fumadores o de bebidas alcohólicas. Por lo tanto, para crear un sistema de verificación de edad un poco más fiable que solo preguntar por la edad al usuario, existen verificaciones basadas en preguntas de cultura general, aunque, no es de los sistemas más fiables que puede encontrar. Por consiguiente, una de las maneras más fiables de que un usuario diga su edad real es pedirle su fecha de nacimiento junto con su color favorito o el nombre de su mascota e indicarle que estos dos datos serán los datos que se le pedirá en caso de que no recuerde su contraseña, por lo tanto, en este caso el usuario aportará, en la mayoría de las veces, sus datos reales.

1.2.2. Implementación del código con varias imágenes

En este apartado, se verá en forma de código los ejemplos que, anteriormente, vio de manera teórica. Por lo tanto, si quiere crear un sitio donde acceder con un usuario y una contraseña, va a implementar el siguiente código HTML y JavaScript:

```
<!DOCTYPE html>

fichero: acceso.html

<html lang="es">

    <head>

        <title></title>

        <script>

            function validarContrasena (passUsuario)
            {

                var patron =/(?!^[0-9]*$)(?!^[a-zA-Z]*$)^([a-zA-Z0-9]{8,20})$/;
                var mensajeError ="La contraseña es incorrecta";

                if ((passUsuario.value.match(patron)) && (passUsuario.value!=""))

                {
                    //alert ('La contraseña es correcta');

                    console.log("La contraseña es correcta");
                }
            }
        </script>
    </head>
    <body>
        <h1>Acceso</h1>
        <form>
            <input type="text" name="usuario" placeholder="Nombre de usuario">
            <input type="password" name="contraseña" placeholder="Contraseña">
            <input type="button" value="Acceder" onclick="validarContrasena(contraseña)">
        </form>
    </body>
</html>
```

```
        }

    else {

        alert (mensajeError);

        passUsuario.focus ();

    }

}

</script>

</head>

<body>

<form action = "verificarContrasena.php" method="get">

    <label for="user">Usuario</label>

    <input type="text" name="user" id="user">

    <label for="pass">Contraseña</label>

    <input type="password" name="pass" id="pass" onblur="validarContrasena(this);">

    <input name="button" type="submit" value="Acceder">

    <br/>

    La contraseña debe tener entre 8 y 20 caracteres, debe contener al menos un dígito y no
    puede contener caracteres especiales

</form>

</body>

</html>
```

Por otro lado, va a comprobar en el lado del servidor que los datos recogidos en el lado del cliente son válidos y veraces. Para ello, debe contar con una tabla específica en la base de datos que habrá creado anteriormente a la recolección de datos. Con el siguiente código se creará dicha tabla en lenguaje SQL, que es el lenguaje que manejan las bases de datos. Véala a continuación:

fichero: bbdd1.sql

```
create database db1;

use db1;

create table users (
    id int not NULL AUTO_INCREMENT,
    name varchar (30) not NULL,
    pass varchar (50) not NULL,
    PRIMARY KEY (id));

insert into users (name, pass) values ("Miguel", "sevilla45");

insert into users (name, pass) values ("Sara", "rockandroll87");

insert into users (name, pass) values ("Lidia", "azulcielo321");
```

Lo próximo que hará es escribir el código PHP que se encargará de consultar los datos de la base de datos al recibir los datos de usuario vía GET y así permitirá saber si el usuario y la contraseña corresponde a un registro real ubicado en la tabla de usuarios de la base de datos. Vea a continuación el código PHP:

Fichero: verificarContrasena.php

```
<?php

//Datos de acceso a su base de datos

//Hay que sustituir 123 por la contraseña real de la BBDD

$mysqli = new mysqli ("localhost", "root", "123", "bd1");

//Se capturan los parámetros pasados por url mediante GET

verificarContrasena.php?name=$name&pass=$pass

$nameUser = $_GET['user'];

$passUser = $_GET['pass'];

//Ahora, se forma la consulta SQL para buscar $nameUser
```

```
$consulta = "SELECT name, pass FROM users WHERE name='".$nameUser."' limit1";  
  
$resultado = $mysqli ->query($consulta);  
  
$fila = $resultado->fetch_assoc();  
  
//Si la fila = a NULL es porque el nombre de usuario no existe  
  
If ($fila != null){  
  
//Si la contraseña guardada en $passUsuario coincide con la que está guardada en //la tabla, se  
permite el acceso  
  
If($passUsuario == $fila['pass'])  
  
Echo "La contraseña es correcta";  
  
Else  
  
Echo "La contraseña es incorrecta";}  
  
Else  
  
Echo "El nombre de usuario no se ha encontrado";
```

Creación de captcha aritmético simple

Para implementar este tipo de captcha, va a generar dos números aleatorios desde el lado del servidor que van a ser verificados doblemente. Primero, serán verificados mediante JavaScript en el lado del cliente y, después, se verificarán en el lado del servidor. Va a hacer de esta manera por si se diese el caso en el que JavaScript estuviera desactivado del navegador o por si un robot intentaría saltarse la verificación en el lado del cliente.

Lo primero que va a hacer es saber cómo generar dos números aleatorios en el lado del servidor. Para realizar esto, va a usar la función de php rand(a,b), la cual genera un número entero aleatorio comprendido entre los valores a y b. Lo próximo que hará es escribir la siguiente línea que genera y escribe en el código HTML un número aleatorio entero entre 1 y 20. Vea el ejemplo a continuación:

```
<?php echo rand (1,20)?>
```

En este ejemplo que va a realizar, lo único que incluirá el formulario es el propio captcha, sin embargo, en un ejemplo real tendrá, como es coherente, muchos más campos. Como ya sabe, la verificación que realmente aporta seguridad en la veracidad de los datos, es la que se realiza en el lado del servidor, sin embargo, realizará una verificación antes en el lado del cliente para evitar que se recargue toda la página en caso de error en los datos y, por consiguiente, le aporta al usuario una mejor experiencia de usuario. A continuación, vea el aspecto que tendrá su formulario:

Resuelva el captcha aritmético

¿Es usted un humano?

[11] + [8] = []

[Enviar]

Al crear este formulario captcha, debe asegurar de que los dos números aleatorios dados no sean modificados por el usuario. Para evitar esto, debe crear un código que solo permita la lectura de los mismos. Vea el ejemplo a continuación:

```
<input id="num1" class="sum" type="text" name="num1" value=<?php echo rand(1,20)?>" readonly="readonly"/>+<input id="num2" class="sum" type="text" name="num2" value=<?php echo rand(1,20)?>" readonly="readonly"/>=
```

Dentro del formulario, creará una capa a la que le asigna el id de mensaje. Esta capa va a permitir interactuar con el usuario y en caso de que el usuario ponga como resultado de la suma un número incorrecto, se le mostrará un mensaje de error antes de que el usuario envíe el formulario y este sea interceptado por el *script* que tiene en el lado del servidor. Vea el ejemplo de código:

```
<h1>Ejemplo de captcha aritmético</h1>

<div id="main">

<div id="mensaje">¿Es usted humano?</div>

<div class="formContainer">

<form name="captcha" id="captchaForm" method="post" action="vCaptcha.php">

<input id="num1" class="sum" type="text" name="num1" value=<?php echo rand(1,20)?>" readonly="readonly"/>+<input id="num2" class="sum" type="text" name="num2" value=<?php echo rand(1,20)?>" readonly="readonly"/>=

<input id="captcha" type="text" name="captcha"/>

<input id="eCaptcha" type="submit" value="Enviar"/>

</form>

</div>

</div>
```

Para mejorar el estilo de su formulario, añade un trozo de código CSS. Esto ayudará a mejorar la experiencia de usuario y el atractivo de su formulario. Véalo a continuación:

```
#main {margin: 5px; border: 1px solid black; width: 350px}

.sum, #captcha {width: 30px}

#num1 {margin-right: 5px}

#num2, #captcha {margin: 5px}

.formContainer, #mensaje {padding: 5px 20px; font: bold 12px/16px Arial, Serif}

input[type=submit] {margin-left: 50px; width: 100px}
```

Lo próximo que hará es definir una función que se llamará `captchaOk()`. Esta función retornará `true` si el usuario resuelve el captcha correctamente y `false` si el resultado no coincide con la suma de ambos números generados. Para hacer esto, debe utilizar el método `getElementById` del objeto `document` que deberá tomar el id del campo como parámetro. A continuación, vea el código de ejemplo:

```
function captchaOk()

{
    var num1 = parseInt(document.getElementById('num1').value);

    var num2 = parseInt(document.getElementById('num2').value);

    var result = parseInt(document.getElementById('captcha').value);

    return ((num1+num2) == result);
}
```

Ahora, creará una instrucción para asegurar que se cargue la página correctamente y, una vez se cargue la página completamente, definirá una función anónima que se ejecutará cuando el usuario clique en el botón Enviar. Véalo en el siguiente ejemplo:

```
window.onload = function ()

{
    document.getElementById('eCaptcha').onclick = function (e)
```

```
If (!captchaOk())  
{  
    e.preventDefault();  
  
    document.getElementById('mensaje').innerHTML = "Vuelva a intentarlo";  
}  
};  
}
```



TOME NOTA

La función anónima se define como function(e) para poder detener la propagación del evento en el caso de que el resultado del captcha sea incorrecto.

Además, al utilizar la función e.preventDefault() evita que el formulario se envíe y, en su lugar, se muestre el mensaje Vuelva a intentarlo.

Para finalizar, va a mostrar en el lado del servidor que el usuario ha completado el captcha con éxito. Lo que va a hacer es capturar los datos del formulario utilizando el método post, y calcular que el resultado es correcto mostrando el mensaje correspondiente. Vea el ejemplo siguiente:

```
<?php  
  
$num1 = isset ($_POST['num1'])? $_POST['num1']: "a";  
  
$num2 = isset ($_POST['num2'])? $_POST['num2']: "b";  
  
$total = isset ($_POST['captcha'])? $_POST['captcha']: "t";  
  
If (($num1+ $num2) == $total)  
  
Echo "Captcha completado con éxito";  
  
Else  
  
Echo "Captcha incorrecto";
```

?>



TOME NOTA

Recuerde que este tipo de captchas se utilizan en entornos reales y que, en lugar de números, también se pueden crear de otros tipos como, por ejemplo, que el usuario escoja una palabra de un conjunto. Por ejemplo, escoja la palabra que no es un color del siguiente conjunto azul, amarillo, rojo, coche.

Para concluir este apartado, va a ver un ejemplo de cómo realizar la validación y la verificación de los datos al mismo tiempo en que son introducidos por el usuario. Para ello, se usa JavaScript y AJAX. Puesto que AJAX permite leer datos del servidor de manera asíncrona lo que permite que no haya que esperar respuesta del servidor, es por ello que no habrá la necesidad de recargar la página para recibir la respuesta del servidor.

En el siguiente ejemplo, utiliza la librería de JQuery, ya que, permite simplificar la manera en que interactúa con el código HTML, además, también puede modificar el árbol DOM y puede utilizar AJAX. Para incluir JQuery, recuerde que debía introducir la siguiente línea de código en su documento HTML:

```
<script src="js/jquery-1.11.0.min.js"></script>
```

Para que la librería JQuery funcione, debe descargar su última versión en su equipo o, por el contrario, utilizar la versión en remoto que está alojada en Google Hosted Libraries.

En los ejemplos de antes en los cuales no usaba ninguna librería de JavaScript, ha estado utilizado el método `document.getElementById()` para seleccionar los elementos del DOM. Sin embargo, a partir de ahora que va a utilizar JQuery, va a ser mucho más fácil interactuar con su archivo HTML. Vea el ejemplo dando como base el siguiente código:

```
<div class="container" id="contenedor1">Elemento 1</div>
```

```
<div class="container" id="contenedor2">Elemento 2</div>
```

Para seleccionar estos contenedores que pertenecen a la clase container, añade la siguiente línea de código JavaScript:

```
Var classContainer=document.getElementsByClassName('container')
```

También, se puede seleccionar de manera individual el primer elemento añadiendo el siguiente código:

```
Var idContenedor1 = document.getElementById('contenedor1')
```

Si utiliza JQuery puede hacer lo mismo, pero de manera mucho más sencilla. Vea el ejemplo análogo en JQuery:

```
//Retornar un array de elementos que pertenecen a la clase container
```

```
Var classContainer=$('.container')
```

```
//Retornar el primer elemento
```

```
Var idContenedor1 = $('#contenedor1')
```

De aquí en adelante, puede utilizar cada uno de los recursos de la librería JQuery para manipular el DOM como quiera. Por ejemplo, vea cómo puede cambiar el texto que contiene el contenedor 2:

```
$('#contenedor2').html('Texto contenedor 2')
```

Por el contrario, si quiere ocultar el texto utiliza la función hide(). Vea el ejemplo:

```
$('#contenedor2').hide()
```

Recuerde que antes de utilizar cualquier función de JQuery, debe cerciorarse de que el documento se haya cargado completamente. Por lo tanto, todo el código debe estar dentro de las llaves \$(document).ready(). Vea el ejemplo:

```
$(document).ready()(function(){
```

```
//Código JQuery});
```

1.2.3. Comprobación de los datos introducidos por el usuario

Como ha visto anteriormente JQuery hace una gran implementación de AJAX, lo que facilitará mucho su uso. A continuación, va a ver cómo hacer una llamada a AJAX con JQuery. Vea el ejemplo:

```
$.ajax{
```

```
Type:'GET' o 'POST',
```

```
url: urlRespuesta.php,
```

```
data: {parametro1: 'valor1', 'parametro2: 'valor2',...},
```

```
dataType: 'xml', 'json', 'script', o 'html',
```

```
success: function (respuesta){
```

```
//Mostrar o procesar el resultado almacenado en el array de respuesta  
}  
});
```

Comprobar entradas de usuario con JQuery

El ejemplo más común que puede ver en este apartado es cuando registra una nueva dirección de correo electrónico y al escribir el correo, puede ver instantáneamente antes de enviar el formulario, si esa dirección de correo está cogida o no.

Para realizar de manera práctica este ejemplo, definirá un formulario de registro de usuarios. Para ello, se utiliza un email para el nombre de usuario, que se deberá validar y verificar antes de grabar este dato en su base de datos. También, se definirá un campo para la contraseña que deberá contener números y letras y, también, deberá cumplir como requisito una longitud mínima de 6 caracteres. Para confirmar la contraseña, se obligará al usuario a repetirla en un campo adicional, para verificar que el usuario conoce la contraseña que ha elegido al cien por cien. Por último, añadir que el correo electrónico para que sea válido, deberá validarse y verificarse y que no sea un correo electrónico que esté ya registrado. El formulario que va a diseñar tendrá el siguiente aspecto:

Registro de usuarios

Nombre de usuario*:

Contraseña:

Repetir contraseña:

*El nombre de usuario debe ser una dirección de correos válida que no haya sido registrada con anterioridad

[Confirmar registro]

El primer paso será programar el CSS, el cual permitirá tener un estilo atractivo para el usuario. Vea el código de ejemplo:

```
<style type="text/css">
```

```
Body{
```

```
Font: 14px/16px Verdana, Sans-Serif;
```

Background: lightblue

}

H1{margin-bottom: 40px}

#container{

Width: 440px;

Margin: 50px auto;

Padding: 40px;

Border: 1px solid lightgray;

Background: White

}

Form label {width: 300px}

Form input [type="text"], form input[type="password"]

{width: 160px}

Form input [type="submit"]

{

Text-align: center;

Width: 160px;

Height: 40px;

Font-weight: 800;

Background-color: lightblue;

Border: 1px solid black

}

Form input [type="submit"]:hover{

```
Cursor: pointer;  
  
Background-color: yellow  
  
}  
  
Form .line{  
  
Clear: both;  
  
Height: 40px;  
  
Margin-bottom: 10px  
  
}  
  
.line div{  
  
Float: left;  
  
Width: 170px;  
  
Margin-left: 15px  
  
}  
  
.mensaje {  
  
Width: 90% !important;  
  
Min-height: 30px;  
  
Padding: 5px;  
  
Margin-bottom: 10px;  
  
Border-bottom: 1px solid lightblue  
  
}  
  
B{  
  
Font-weight: 800;  
  
Text-decoration: underline
```

```
}
```

```
.clear {clear: both}
```

```
</style>
```

Debe saber que la clase .mensaje va a definir las áreas donde se mostrarán los mensajes de validación y verificación de cada campo. Para mostrar esos mensajes, se usa el método de JQuery .html('text') para el cual debe seleccionar el identificador correspondiente del div donde quiera mostrarlo. Vea el ejemplo dónde se definen esas áreas:

```
<div class="mensaje" id="mensajeUsuario"></div>
```

```
<div class="mensaje" id="mensajePass"></div>
```

```
<div class="mensaje" id="mensajeRpass"></div>
```

Lo próximos que va a hacer es programar el código HTML de su formulario. Véalo a continuación:

```
<div id="container">
```

```
<form action="registroUsuario.php" method="post" name="registro" id="formularioRegistro">
```

```
<div>
```

```
<h1>Registro de usuarios</h1>
```

```
<div class="line">
```

```
<div><label for="username">Nombre de usuario*</label></div>
```

```
<div><input type="text" name="username" id="username"/></div>
```

```
<div class="clear"></div>
```

```
<div class="mensaje" id="mensajeUsuario"></div>
```

```
</div>
```

```
<div class="line">
```

```
<div><label for="pwd1">Contraseña:</label></div>
```

```
<div><input type="password" name="pwd1" id="pwd1"/></div>
```

```
<div class="clear"></div>
```

```
<div class="mensaje" id="mensajePass"></div>

</div>

<div class="line">

<div><label for="pwd2">Repita la contraseña:</label></div>

<div><input type="password" name="pwd2" id="pwd2"/></div>

<div class="clear"></div>

<div class="mensaje" id="mensajeRpass"></div>

</div>

<div class="clear"></div>

<div><p>*El nombre de usuario debe ser una dirección de correos válida que no haya sido registrada.</p>

</div>

<input type="submit" value="Confirmar registro"/>

</div>

</form>

</div>
```

El formulario que ha creado, enviará los datos al fichero registroUsuario.php, que será el encargado de guardar los datos que ha introducido el usuario en su base de datos. Como se ha dicho anteriormente, la validación y la verificación de los datos se realizaran al mismo tiempo en el que el usuario introduzca los datos en los campos del formulario. Vea el siguiente ejemplo de código:

```
Var key = window.event ? evento.keyCode: evento.which;
```

Esta línea de código permite capturar los códigos ASCII y realizar un evento. Sin embargo, esta vez se hará de otra manera, ya que, si utiliza esa línea de código, no captará los eventos si el usuario tiene en su navegador que se rellenen los campos de manera automática o si, por ejemplo, utiliza para pegar el texto las teclas CTRL + V. Por lo tanto, para evitar errores, se utiliza el método setInterval del objeto window, así podrá hacer que JavaScript ejecute una función de manera periódica. En esta función el parámetro de tiempo definirá cada cuántos milisegundos se llama a la función. Vea el siguiente ejemplo donde se muestra el uso de esta función.

setInterval (cambiosFormulario, 0.5);

definiremos dos objetos de datos:

```
var entradaFormulario = {username: "", pwd1:"", pwd2:""}
```

```
var resFormulario = {username: false, pwd1: false, pwd2: false}
```

En la variable entradaFormulario se almacenará siempre los últimos valores de los campos del formulario, en cambio, en la variable resFormulario se almacenará el estado del proceso de validación y verificación. Es decir, cuando resFormulario cambie de *false* a *true*, sabrá que el formulario es correcto y estará listo y se enviarán los datos a la variable entradaFormulario.

Se ha programado que cada medio milisegundo se ejecute la función cambiosFormulario(). Cada vez que se ejecute, esta recorrerá el objeto entradaFormulario y comprobará si alguna entrada se ha modificado. Si alguna entrada ha sido modificada, se almacenarán los nuevos datos en las variables entradasFormulario y resFormulario y se mostrarán los mensajes correspondientes. Vea el trozo de pseudocódigo de ejemplo.

```
Function cambiosFormulario()
```

```
{
```

```
Entrada = (cada entrada de entradasFormulario)
```

```
{
```

```
nuevaEntrada = valor actual input
```

```
SI (nuevaEntrada != entradasFormulario)
```

```
{
```

```
Guardar valor actual de input en entradasFormulario;
```

```
Validar nuevaEntrada
```

```
Verificar nuevaEntrada
```

```
Mostrar mensajes (ok/kao)
```

```
Actualizar resFormulario a true si
```

```
nuevaEntrada es validada y verificada
```

```
} //fin SI
```

}

Lo próximo que va a ver es cómo asignar el evento submit a una función con la que se asegurará que los datos introducidos por el usuario son correctos. Véalo a continuación:

```
resFormulario = {true, true, true}

$('#formRegistro').on('submit', function(e)

{

If (resFormulario.username && resFormulario.pwd1 && resFormulario.pwd2)

{

//Datos validados y verificados

This.submit();

}

else

//Si hay errores, no se permite enviar el formulario

e.preventDefault();

//Mostrar errores de validación

});
```

A continuación, verá cómo implementar la función cambiosFormulario(), para ello, va a enumerar cada uno de los elementos del objeto entradasFormulario. Vea el ejemplo:

```
For (var entrada in entradasFormulario){...}
```

Ya que, el objeto entrada apunta a cada elemento de entradasFormulario, va a dirigir mediante JQuery los elementos de entradasFormulario y los identificadores de los inputs del formulario de la siguiente manera:

```
$("#" +entrada)
```

Para poder saber si desde la última vez que se ha ejecutado la función cambiosFormulario() ha habido algún cambio en los datos del formulario, debe guardar el valor en una variable llamada nuevaEntrada = \$('#'+entrada).val() y comparar con el valor anterior introduciendo el siguiente código:

```
If(nuevaEntrada != entradasFormulario [entrada]){...}
```

Si ha cambiado el contenido de la función, esta condición será cierta, por lo que, se guardará el nuevo valor en entradasFormulario y se comprobará si ha habido algún cambio en la verificación y validación del formulario. Se llamará a la función correspondiente dependiendo del campo que haya sido modificado, esta función actualizará resFormulario y los mensajes informativos. Para realizar esta acción utiliza *switch case* de la siguiente manera:

```
Switch (entrada)
```

```
{
```

```
Case: "username":
```

```
validarUsuario (nuevaEntrada);
```

```
verificarUsurio (nuevaEntrada);
```

```
break;
```

```
case "pwd1":
```

```
validarPass (nuevaEntrada);
```

```
break;
```

```
case "pwd2":
```

```
verificarPass (nuevaEntrada);
```

```
break;
```

```
}
```

Vea el código completo de la función en JavaScript:

```
Function cambiosFormulario ()
```

```
{
```

```
For (var entrada in entradasFormulario)
```

```
{
```

```
Var nuevaEntrada = $('#'+entrada).val();
```

```
If(nuevaEntrada != entradasFormulario [entrada])  
{  
    entradasFormulario [entrada] = $('#'+entrada).val();  
  
    switch (entrada)  
    {  
        case "username":  
            if (validarUsuario (nuevaEntrada))  
            {  
                verificarUsuario();  
            }  
  
        Else  
        {  
            $('#mensajeUsuario').html("");  
            $('#pwd1').val(""); resFormulario.username=false;  
        }  
  
        Break;  
  
        Case "pwd1":  
            $('#pwd2').val("");  
            If(nuevaEntrada == "")  
            {  
                $('#mensajePass').html("");  
                $('#mensajeRpass').html("");  
                resFormulario.pwd1=false;  
            }  
    }  
}
```

}

Else if (validarPass (nuevaEntrada))

{

\$('#mensajePass').html('<b class="mensajeOk">Contraseña fuerte');

resFormulario.pwd1=true;

}

Else

{

\$('#mensajePass').html('<b class="mensjaeOk">Contraseña débil');

\$('mensajeRpass').html(""); resFormulario.pwd1=false;

resFormulario.pwd2=false;

}

Break;

Case "pwd2":

If ((entradasFormulario ['pwd1'] == "") || !validarPass(entradasFormulario['pwd1']))

{

\$("#pwd2").val("");

resFormulario.pwd2=false;

}

Else if (verificarPass (entradasFormulario ['pwd1'], entradasFormulario ['pwd2']))

{

\$('#mensajeRpass').html('<b class="mensajeOk">Las contraseña coinciden');

resFormulario.pwd2=true;

```
}

Else

{

$'#mensajeRpass').html('');

resFormulario.pwd2 = false;

}

Break;

}

}

}

}

}
```

La función validarUsuario() funciona de igual manera que la función que servía para validar dirección de correo electrónico que se ha visto en apartados anteriores. De la misma manera, también se vio anteriormente la función que valida la contraseña. Se utiliza la función verificarPass() para verificar si ambas contraseñas coinciden. Esta función retornará un valor true si ambas contraseñas coinciden y, retornará un valor *false* en el caso contrario. Véala a continuación:

```
Function verificarPass (pass1, pass2){  
    Return (pass1 == pass2);  
}
```

El último paso que quedaría dar es la verificación del nombre de usuario. Para realizar esta verificación, se utiliza la función de JQuery \$.ajax que enviará los datos vía POST al archivo verificarUsuario.php. Además, se define en un archivo Json el tipo de datos que se enviarán y una función callback que se ejecutará cuando el servidor proporcione una respuesta. Se pone la función *callback* en success: function (json){...}.

Por otro lado, los datos a verificar se definirán en data: {...}. Estos datos se definen mediante un objeto y en este caso el objeto tan solo contiene un elemento que será username: entradasFormulario.username. Vea a continuación el código donde se implementa la función verificarUsuario():

```
Function verificarUsuario(){
```

```
$.ajax({  
    type:'POST',  
    url: 'verificarUsuario.php',  
    data: {username: entradasFormulario.username},  
    dataType: 'json',  
    success: function(json) {if(json.usuarioNoExiste){  
        $('#mensajeUsuario').html('<b class="mensajeOk">Nombre de usuario disponible</b>');  
        resFormulario.username=true;  
    }  
    Else  
        $('#mensajeUsuario').html('<b class="mensajeOk">Nombre de usuario no disponible</b>');  
    }  
});  
}
```

Como ha visto anteriormente, cuando esta función se ejecuta llama al fichero verificarUsuario.php y le pasa como parámetro el objeto que verá continuación:

Data: {username: entradasFormulario.username}.

Vea el código de ejemplo del fichero verificarUsuario.php a continuación:

Fichero verificarUsuario.php

```
<?php  
  
$mysqli = new mysqli ("localhost", "root", "1234", "ejemplo");  
  
$nombreUsuario = $_POST['username'];  
  
$consulta = "SELECT name FROM users WHERE name='".$nombreUsuario."' limit1";  
  
$result = $mysqli -> query($consulta);
```

```
$fila = $result -> fetch_assoc();  
  
$usuarioNoExiste = ($fila === null);  
  
$datos = array ("usuarioNoExiste" => $usuarioNoExiste);  
  
Echo json_encode ($datos);  
  
?>
```

Si al realizar la consulta no obtiene resultado, significa que el nombre de usuario no ha sido previamente registrado. En ese caso, se devuelve el resultado en \$datos = array("usuarioNoExiste" => \$usuarioNoExiste). Si el nombre de usuario no se ha utilizado antes y, por lo tanto, se ha verificado, \$usuarioNoExiste será igual a true. La última línea del ejemplo anterior la cual es json_encode(\$datos) es la encargada de escribir el resultado como matriz de JavaScript con el formato siguiente json={"usuarioNoExiste":true} o json={"usuarioNoExiste":false}. Cuando se envía el formulario, que será cuando todos los datos hayan sido validados y verificados, se llama al fichero registroUsuario.php. Este fichero es el encargado de insertar una nueva fila en la tabla users de la base de datos con los datos de usuario del nuevo registro. A continuación, vea el ejemplo de cómo se estructura el archivo registroUsuario.php tan solo por curiosidad, ya que no hace falta que entienda su funcionamiento:

Fichero: registroUsuario.php

```
<!DOCTYPE html>  
  
<html Lang=es>  
  
<head>  
  
<meta charset=Windows-1252>  
  
<title>Registro de usuario</title>  
  
<style type="text/css">  
  
Body {Font: 14px/16px Verdana, Sans-Serif; background: lightblue}  
  
H2{background-color:lightgray; color:blue}  
  
.container {margin: 50px auto; border:1px solid black; padding: 30px; background-color:lightgray;  
  
Width: 600px; height: 200px}  
  
A, a:hover, a:active, a:visited{color:black}
```

```
</style>

</head>

<body>

<?php

$mysqli = new mysqli ("localhost","root","123","adwe");

$nombreUsuario = $_POST['username'];

$pass = $_POST['pwd1'];

$consulta="insert into users (name, pass) values('".$nombreUsuario."','".$.$pass."');"

$result = $mysqli -> query ($consulta);

?>

<div class="container">

<h1>Usuario registrado con éxito</h1>

<h2><?php echo($nombreUsuario);?></h2>

<a href="registro.html">Registrar otro usuario</a>

</div>

</body>

</html>
```

Vea a continuación el código completo del ejemplo del formulario de registro:

```
<!DOCTYPE html>

<html Lang=es>

<head>

<meta charset=Windows-1252>
```

```
<title>Registro de Usuario</title>

<style type="text/css">

<body{Font: 14px/16px Verdana, Sans-Serif; background: lightblue} h1 {margin-bottom: 40px}

#container {width: 440px; margin: 50px auto; padding: 40px; border: 1px solid lightgray; background: blue}

Form label {width: 300px}, form input[type="text"], form input [type="password"] {width:160px,}

Form input [type="submit"] {text-align: center; width: 160px; height:40px; Font-weight: 800;

Background-color: lightblue; border: 1px solid black}

Form input [type="submit"]:hover {cursor: pointer; background-color: yellow}

Form .line{clear: both; height: 40px; margin-bottom: 10px}

.line div {float: left; width: 170px; margin-left: 15px}

.mensaje {width: 90% !important; min-height: 30px; padding: 5px; margin-bottom: 10px; border-bottom: 1px solid lightblue}

B {Font-weight: 800; text-decoration: underline}

.clear {clear:both}

</style>

<script src="//Ajax.googleapis.com/Ajax/libs/jquery/1.11.0/jquery.min.js"></script>

<script>

Var entradasFormulario = {username: "", pwd1: "", pwd2:""}

Var resFormulario = {username: false, pwd1: false, pwd2: false}

Function borrarFormulario(){

$('#mensajeUsuario').html("");

$('#mensajePass').html("");

$('#mensajeRpass').html("");



```

```
$(‘#username’).val(“);  
$(‘#pwd1’).val(“);  
$(‘#pwd2’).val(“);  
$(“input: text:visible:first”).focus();  
}  
$(documento).ready(function()  
{  
borrarFormulario();  
function validarUsuario (emailUsuario)  
{  
var patron=/^([a-zA-Z0-9_\.\\-])+\\@(([a-zA-Z0-9\\-])+\\.)+([a-zA-Z]{2,6}+$/;  
vare Regular = new RegExp(patron);  
return (eRegular.test(emailUsuario) && (emailUsuario.value!=”))  
}  
Function verificarUsuario()  
{  
$.ajax({  
Type:’POST’,  
url: ‘verificarUsuario.php’,  
data: {username: entradasFormulario.username},  
dataType:’json’,  
success: function(json)  
{
```

```
if(json.usuarioNoExiste)
{
    $('#mensajeUsuario').html('<b class="mensajeOk">Nombre de usuario disponible</b>');
    resFormulario.username = true;
}

Else
{
    $('#mensajeUsuario').html('<b class="mensajeOk">Nombre de usuario no disponible</b>');
}
});

}

Function validarPass (passUsuario)
{
    Var patron = /(!^([0-9]*$)(!^[a-zA-Z]*$)^([a-zA-Z0-9]{6,20})$/;
    Var eRegular = new RegExp (patron);
    Return (eRegular.test (passUsuario) && (passUsuario.value!=""));
}

Function verificarPass (pass1, pass2)
{
    Return (pass1 === pass2);
}

$('#registroForm').on('submit', function (e)
{
    If (resFormulario.username $$ resFormulario.pwd1 && resFormulario.pwd2)
```

```
//alert ("Usuario registrado con éxito");

This.submit();

}

Else

{

//No permite enviar el formulario

e.preventDefault();

if(!resFormulario.username)

{

$('#mensajeUsuario').html('<b class="mensajeOk">Debe introducir un nombre de usuario</b>');

$('#mensajePass').html("");

}

Else if (!resFormulario.pwd1)

{

$('#mensajePass').html('<b class="mensajeOk">Debe introducir una contraseña válida</b>');

$('#mensajeRpass').html("");

}

Else if (!resFormulario.pwd2)

If (entradasFormulario.pwd2 == "")

$('#mensajeRpass').html('<b class="mensajeOk">Debe repetir la contraseñas</b>');

Else{

$('#mensajeRpass').html('<b class="mensajeOk">Las contraseñas no coinciden</b>');

$('#pwd2'.val("");



```

```
}

}

});

cambiosFormulario()

{

for (var entrada in entradasFormulario)

{

var nuevaEntrada = $('#'+entrada).val();

if(nuevaEntrada != entradasFormulario [entrada])

{

entradasFormulario [entrada] = $('#'+entrada).val();

switch (entrada)

{

Case "username":


If(validarUsuario (nuevaEntrada))

{

verificarUsuario();

}

Else

{



$('#mensajeUsuario').html("");


$('#pwd1').val("")();


resFormulario.username = false;
```

}

Break;

Case “pwd1”:

\$('#pwd2').val("");

If (nuevaEntrada == "")

{

\$('#mensajePass').html("");

\$('#mensajeRpass').html("");

resFormulario.pwd1 = false;

}

Else if (validarPass (nuevaEntrada))

{

\$('#mensajePass').html('<b class="mensajeOk">Contraseña fuerte');

resFormulario.pwd1 = true;

}

Else

{

\$('#mensajePass').html('<b class="mensajeOk">Contraseña débil');

\$('#mensajeRpass').html(""); resFormulario.pwd1 = false; resFormulario.pwd2 = false;

}

Break;

Case “pwd2”:

If ((entradasFormulario ['pwd1'] == "") || !validarPass(entradasFormulario ['pwd1']))

```
{  
    $('#pwd2').val("");  
  
    resFormulario.pwd2 = false;  
  
}  
  
Else if (verificarPass (entradasFormulario ['pwd1'], entradasFormulario ['pwd2']))  
  
{  
  
    $('#mensajeRpass').html('<b class="mensajeOk">Las contraseñas coinciden</b>');  
  
    resFormulario.pwd2 = true;  
  
}  
  
Else  
  
{  
  
    $('#mensajeRpass').html("");  
  
    resFormulario.pwd2 = false;  
  
}  
  
Break;  
  
}  
  
}  
  
}  
  
}  
  
setInterval(cambiosFormulario, 0.5);  
  
});  
  
</script>  
  
<body>
```

```
<div id="container">

<form action="registroUsuario.php" method="post" name="registro" id="registroForm">

<div>

<h1>Registro de usuario</h1>

<div class="line">

<div><label for="username">Nombre de usuario*</label></div>

<div><input type="text" name="username" id="username"/></div>

<div class="clear"></div>

<div class="mensaje" id="mensajeUsuario"></div>

</div>

<div class="line">

<div><label for="pwd1">Contraseña:</label></div>

<div><input type="password" name="pwd1" id="pwd1"/></div>

<div class="clear"></div>

<div class="mensaje" id="mensajePass"></div>

</div>

<div class="line"><div><label for="pwd2">Repite la contraseña:</label></div>

<div><input type="password" name="pwd2" id="pwd2"/></div>

<div class="clear"></div>

<div class="mensaje" id="mensajeRpass"></div>

</div>

<div class="clear"></div>

<div>
```

```
<p>*El nombre de usuario debe ser una<b> dirección de correos válida </b>que no haya sido registrada.</p>

</div>

<input type="submit" value="Confirmar registro"/>

</div>

</form>

</div>

</body>

</html>
```

2. EFECTOS ESPECIALES EN PÁGINAS WEB

En este tema se verá cómo crear efectos especiales en páginas web. Estos efectos ayudan a hacer la interfaz de usuario más atractiva y, por lo tanto, mejoran la presencia de su web. Para lograr crear estas animaciones, existen diferentes herramientas que ayudan a generar páginas más dinámicas.

Los efectos visuales pueden ser tanto de animaciones en los componentes de las webs como, por ejemplo, algo tan simple como cambiar el color de las letras cuando pasa el ratón por encima de la sección donde están ubicadas. Cabe destacar que, es igual de poco atractivo hacer una web sin efectos, como recargar excesivamente la web con efectos.

Podría clasificar los efectos especiales de las webs en efectos especiales a gran escala que son los que utiliza para generar mayor impacto en el usuario, que podrían ser, por ejemplo, efectos de scroll parallax y las notificaciones emergentes, y en efectos especiales a pequeña escala, que son los que no necesitan que el usuario interactúe con ellos. Estos efectos pueden ser, por ejemplo, las barras de carga de la web o añadir una imagen animada.

Si parte de la clasificación anterior, tiene las siguientes animaciones:

- **Animaciones de carga.** Estas animaciones se muestran cuando la página se está cargando y, con ellas, puede entretenir a los usuarios mientras esperan. La animación más típica de carga es como la que vea a continuación:



- **Non Scrolling.** Son animaciones que ahorran espacio en pantalla, ya que son menús de navegación que se encuentran ocultos hasta que el usuario mueve el ratón sobre ellos.
- **Hover.** Este es uno de los efectos más sencillos y más utilizados. Este efecto ocurre cuando se pasa el ratón por encima de un elemento y este cambia de color o forma. Normalmente, se utiliza en botones y en textos de enlace.
- **Slideshows.** Este efecto es muy útil cuando quiere mostrar varias imágenes a los usuarios y que no les resulte pesado ver el contenido. En los slideshows las imágenes se deslizan unas tras otras.
- **Scrolling.** El scrolling permite que vaya apareciendo el contenido de una web a medida que el usuario se desplace por la propia web.

- **Ilustraciones animadas.** Son animaciones en la web que se utilizan para complementar el contenido de la propia web. En este tipo de efectos no hace falta que el usuario interactúe con el elemento para que este haga su función.
- **Background.** En las webs para hacerlas más dinámicas se pueden emplear fondos animados, pero hay que tener cuidado con esto para no saturar al usuario con excesivas animaciones. Por ejemplo, algo que se suele utilizar muy a menudo es el efecto parallax, que permite simular profundidad en las imágenes.

2.1. Trabajar con imágenes: imágenes de sustitución e imágenes múltiples

Al principio de los tiempos del desarrollo web, las webs consistían en puro texto, lo cual las páginas eran muy poco atractivas para el usuario y totalmente estáticas. Gracias al avance tecnológico, hoy por hoy las páginas web tienden a ser lo más dinámicas posible y, sobre todo, hacer que la interfaz de usuario sea lo más atractiva posible es uno de los puntos más importantes del desarrollo web actual. Es por ello, que en la actualidad tiene mucho peso el uso de imágenes, animaciones, secuencias de vídeo y otros elementos dinámicos que puede añadir a su web. Con estos elementos, puede aportar color, movimiento y a veces, interactividad a su web. Es sabido que la mayoría de usuarios cuando acceden a una web se sienten atraídos por cómo de atractiva es la interfaz de usuario. Este atractivo se pone mediante el uso de imágenes, ya que, pueden ser elementos decorativos que pueden aparecer en el fondo de algún texto, o ser parte de la información que está dando su web, donde muchas veces una imagen aporta mayor información que el propio texto. Por ejemplo, las situaciones donde una imagen aporta más información que un texto puede ser un esquema de diseño, un gráfico, el plano de un edificio, etc.

En este epígrafe aprenda cómo utilizar todo lo que aporta HTML para hacer sus webs más atractivas pudiendo añadir imágenes a sus documentos, ya sean en forma de fondo, como objetos que formen parte de la propia web o, por ejemplo, como marcadores para una lista. Para realizar todo esto, debe controlar características como el tamaño, la disposición, los márgenes, entre otros aspectos, mediante el atributo img de HTML y, también, debe controlar un poco de código CSS. Además, también tendrá la posibilidad de utilizar imágenes en forma de hipervínculos, tanto de forma directa, como creando una definición de mapa de imagen.

	<p><i>Las webs además de contener texto, pueden contener elementos multimedia, los cuales aportarán mayor atractivo a su web, como imágenes, vídeos, música, mapas de imágenes e incluso otros documentos en HTML.</i></p>
TOME NOTA	

Para insertar una imagen, lo más común es utilizar el atributo img de HTML. El elemento img permitirá que cuando la página se cargue, se muestre la imagen que ha añadido en el atributo src que, a su vez, pertenece a img. Al utilizar ambos elementos, puede añadir una imagen en un párrafo,

en un elemento de una lista o en un encabezado. Para utilizar img, debe escribir la etiqueta de apertura de forma obligatoria, pero puede prescindir de la etiqueta de cierre, aunque utilizarla supone una buena praxis.

Vea a continuación una lista donde puede observar los atributos que pueden aplicarse a una imagen:

Atributo	Valor	Descripción
Alt	Texto alternativo	Este texto alternativo se muestra en la página cuando la imagen está rota.
Src	url	Aquí es donde se especifica la url donde se encuentra la imagen a insertar
Align	Left Right Top Bottom middle	Especifica la posición de la imagen
Border	Píxeles	Pone un borde a la imagen
Height	Píxeles %	Especifica la altura de la imagen
Hspace	Píxeles	Especifica el espacio en blanco que se debe dejar a la derecha y ala izquierda de la imagen
Ismap	Ismap	Especifica una imagen como mapa de imagen en el lado del servidor
Longdesc	url	Pone un enlace a una descripción larga de la imagen
Usemap	#nombreMapa	Especifica una imagen como mapa de imagen en el lado del cliente
Vspace	Píxeles	Especifica la cantidad de espacio que se debe dejar arroba y debajo de la imagen
Width	Píxeles %	Especifica el ancho de imagen

A continuación, verá una tabla donde puede observar los atributos estándar de la etiqueta img:

Atributos	Valor	Descripción
-----------	-------	-------------

Class	Nombre de la clase	Asigna un nombre de clase.
		Este atributo actúa como selector para hojas de estilo al asignar información de estilo a un conjunto de elementos.
Id	Nombre identificador	Se utiliza como selector para hojas de estilo, como vínculo destino para enlaces de hipertexto, como medio para referenciar a un elemento desde un script, o como nombre de un elemento object declarado
Style	Estilo	Especifica información de estilo
Title	Nombre de título	Especifica un título a un elemento
Dir	Ltr Rtl	Especifica la dirección del texto. Ltr: de izquierda a derecha. Por ejemplo, el español Rtl: de derecha a izquierda. Por ejemplo, el árabe.
Lang	Código de lenguaje	Especifica el idioma de los valores de los atributos y del texto contenido en un elemento. Es útil para: Ayudar a los motores de búsqueda Ayudar a los sintetizadores de voz Ayudar a los verificadores de ortografía y gramática

Ahora, va a ver una tabla donde se exponen los eventos de la etiqueta img:

Evento	Valor	Descripción
Onmouseup	Script	El script se ejecuta cuando el botón del ratón se suelta al estar encima de un elemento
Onmouseover	Script	Se ejecuta el script cuando el ratón se sitúa encima de un elemento
Onmouseout	Script	El script se ejecuta cuando el puntero del ratón sale de un elemento
Onmousemove	Script	Se ejecuta cuando el ratón se está moviendo sobre un elemento
Onmousedown	Script	Se ejecuta cuando el botón del ratón se pulsa estando encima de un elemento
Ondblclick	Script	Se ejecuta cuando se hace doble clic sobre un elemento
OnClick	Script	Se ejecuta cuando hace clic sobre un elemento
Onkeyup	Script	Se ejecuta cuando se suelta una tecla
Onkeypress	Script	Se ejecuta cuando se pulsa y se suelta una tecla
Onkeydown	Script	Se ejecuta cuando se pulsa una tecla

En la especificación estricta de HTML, es natural que no aparezcan los atributos relativos a presentación como, por ejemplo, align, hspace, border o vspace. Para todos estos atributos, existen propiedades de CSS que realizan la misma función.

Siempre que utilice la etiqueta img, debe utilizar el atributo src, que contendrá la url que enlaza con la imagen quiere mostrar y el atributo alt, que debe contener la descripción de la propia imagen. Este texto alternativo se utiliza, por ejemplo, cuando las imágenes no se pueden mostrar al usuario, ya sea, por algún problema o porque el navegador no soporte los estándares del código utilizado. Algunos navegadores muestran en una pequeña ventana flotante el texto contenido en el atributo alt cuando para el puntero del ratón sobre esta. Vea algunos atributos más del tipo img:

Img	Pone una imagen a la web
Map	Se utiliza para crear el mapa de imagen
Área	Inserta una región geométrica en un mapa de imagen
Object	Objeto genérico
Param	Especifica valores para un objeto necesario en un tiempo de ejecución
Embed	Incrusta contenido multimedia que necesitan <i>plug-ins</i>
Noembed	Contenido que se muestra cuando los contenidos multimedia incrustados no tienen soporte
Applet	Pone un applet (Obsoleto)
Iframe	Marco flotante que se muestra un documento HTML externo

A continuación, vea cómo crear un objeto image utilizando la etiqueta img que aporta el HTML estándar. Véalo:

```
`. Las imágenes que se incrustan en el documento HTML se reúnen en la matriz de document.images []. Además, también puede acceder a las imágenes que no tienen ningún atributo name mediante las propiedades del objeto document. Vea un ejemplo a continuación:

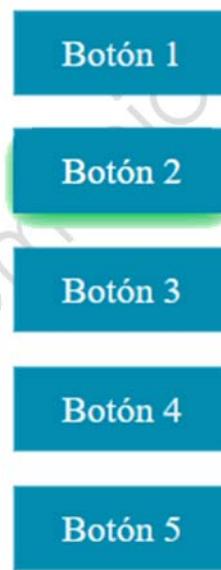
Document.images[0] //Esta sería la primera imagen insertada en la matriz

Document.foto //Así es cómo llamamos a una imagen con el atributo name="foto"

La propiedad más importante del objeto image es src, ya que, cuando añada esta propiedad el explorador carga y modifica la imagen anteriormente especificada por la nueva. Así, es cómo desarrolla los distintos efectos visuales como, por ejemplo, las animaciones y los rollover.

Además, también puede crear de forma dinámica objetos de imagen que no se muestran en pantalla utilizando código JavaScript. Para ello, utiliza la función Image(). Al igual que hacía con las imágenes creadas mediante HTML, con JavaScript también debe indicarle al explorador que muestre una imagen a través de la propiedad src de cualquier otra imagen creada, ya que, no se puede mostrar un objeto image creado de esta manera. Lo único que podrá hacer es obligar al objeto image a descargar una imagen estableciendo la propiedad src de la misma. Aunque no se puedan mostrar los objetos image, son útiles porque estos objetos cargan las imágenes en caché y si utiliza posteriormente la misma url de imagen en una etiqueta img en HTML, las imágenes se mostrarán mucho más rápido, pues ya estaban cargadas en la caché.

Por otra parte, el rollover es un efecto visual que permite que un elemento de la página cambie de estado cuando el puntero del ratón pasa sobre él. Este efecto es uno de los más utilizados, ya que, permite que el usuario sepa que los botones y vínculos son interactivos y, por lo tanto, el usuario puede clicar en ellos. Vea una imagen de ejemplo:



Vea a continuación, cómo conseguir realizar este tipo de efecto con código CSS, HTML y JavaScript. Vea el ejemplo:

```

```

Por otro lado, vea el código CSS que permitirá obtener el efecto que busca:

```
//Imagen original#imagen {
 Backgorund: url(imagen inicial.jpg) no-repeat;
 Height: alto px;
 Width: ancho px;
}

//Imagen de sustitución
#imagen:hover{
 Background: url(imagen sustitución.jpg) no-repeat;
 Height: alto px;
 Width: ancho px;
}
```

Ahora, va a unir ambos códigos, para ver cómo quedaría en la realidad:

```
//HTML

<div id="ejemplo">
</div>

//CSS

#ejemplo {
 Width: anchoimagen px;
 Height: alturaimagen px;
 Background-image: url (src/imagen original);
 Background-repeat: no-repeat;
}

#ejemplo:hover{
```

Background-image: url (src/imagen sustitución);

}

### 2.1.1. Selección de imágenes

Para poder insertar una imagen en su web, antes hay que realizarla, ya sea, dibujando, fotografiando, escaneando, etc. Además, habrá que adecuarla para utilizarla en el espacio en que la precisa, es decir, no es lo mismo poner una imagen de fondo, que ponerla como ícono. Para preparar las imágenes debe utilizar aplicaciones externas como Photoshop o Gimp, aunque hay muchas más. Algunas de estas aplicaciones son más potentes que otras, por lo que, brindarán mejores resultados como, por ejemplo, a la hora de corregir la iluminación de una foto, los colores, eliminar fallos como, por ejemplo, los ojos rojos y aplicar filtros visuales.

Puede utilizar las imágenes de maneras muy diversas:

- **Imagen simple:** se pueden utilizar las imágenes de manera que aporte algún tipo de información, por ejemplo, el logo de una empresa, los banners, fotos de productos, etc.
- **Imagen como vínculo:** se pueden utilizar imágenes que, al pinchar en ellas, transporten a otro documento, este ejemplo se ve mucho en los blogs donde se pone una imagen llamativa sobre un post y al pinchar en ella, lleva al artículo.
- **Imagen como mapa de imagen:** los mapas de imagen permiten tener varias zonas interactivas que vinculan a otros documentos. De este ejemplo, puede ver muchos en las tiendas online donde puede ver imágenes de productos que tienen una zona donde hay un botón integrado en la imagen y al pinchar ahí lleva al producto. Vea el ejemplo en la siguiente imagen:



Normalmente, las imágenes que no aportan ninguna información se destinan a la propia presentación de la página ubicándolas como imagen de fondo. Con CSS podrá insertar imágenes en todas las partes del documento HTML, no solo en el body. Para ello, se utiliza la propiedad background-image.

Al insertar las imágenes decorativas dentro de un documento CSS externo al documento HTML, ofrece muchas ventajas. Además de mantener el documento HTML más limpio y fácil de visualizar, también se hará más sencillo actualizar el archivo en un futuro.

Añadir imágenes a la web es muy importante, pero hay que tener en cuenta que excederse en este aspecto puede correr en contra, ya que, su sitio web tardaría mucho tiempo en cargar debido al gran peso que tienen las imágenes en general. Además, si abusa de las imágenes también le puede ser

difícil al usuario concentrarse en el contenido informativo de su web. Por ello, si quiere un sitio web atractivo, se debe tener un equilibrio entre texto e imágenes.

Como ya sabe, para insertar imágenes en la web, se utiliza la etiqueta img que proporciona HTML. Hay que tener en cuenta que puede insertar imágenes en su web que estén almacenadas en el propio servidor donde se ubica la web o, también, pueden estar almacenadas en otro servidor, eso queda a elección del propio desarrollador.

Según los estándares W3C solo se admiten imágenes con los siguientes formatos:

- **Formato JPEG o JPG.** Este formato se utiliza para imágenes reales como, por ejemplo, fotografías. Esto se debe a que este formato brinda la posibilidad de elegir el nivel de compresión de la imagen para poder obtener una mejor relación entre peso y calidad de la imagen. Este tipo de formato lo que aporta es una gran compresión del archivo, por lo tanto, es ideal para usarlo en las imágenes de su web, ya que, así su web requerirá un menor tiempo de carga. En cambio, este formato se desaconseja si quiere obtener fotografías o imágenes con mucha calidad.
- **Formato GIF.** Este formato se utiliza, sobre todo, para dibujos y no en imágenes reales, ya que, este formato utiliza una paleta de 256 colores y es capaz de contemplar que uno de esos colores sea el color transparente. Este formato, también es utilizado para imágenes animadas. Antiguamente, era el formato más utilizado para las imágenes web, pero al existir un problema con el algoritmo de compresión que utilizaba, se dejó de usar debido a la incompatibilidad.
- **Formato PNG.** Este formato fue diseñado para sustituir el formato GIF debido al problema que se ha comentado anteriormente que tuvo el formato GIF. El formato PNG es un formato que sirve tanto para imágenes reales como para dibujos, ya que, ofrece la posibilidad de controlar la transparencia o la corrección de gamma y, además, ofrece un tipo de compresión de imágenes donde no se pierde calidad. Este formato está aceptado por los navegadores más conocidos, pero puede encontrar aún navegadores donde no es compatible este tipo de formato. Las imágenes que utilizan este formato pueden tener un grado de mayor compresión si estas utilizan un número reducido de colores. Este formato, también, permite crear imágenes que aparecen una tras otra progresivamente e imágenes donde existe el color transparente.

Aunque existen más tipos de formato de imágenes, estos tres formatos están totalmente aceptados por los navegadores más conocidos, por lo tanto, serán los formatos que utilice a la hora de crear una página web.

Por otra parte, si quiere utilizar una imagen de fondo, debe recordar que no debe tener colores muy intensos ni brillantes, puesto que, dificultará la lectura del contenido de su web. Para ello, puede utilizar una aplicación de diseño gráfico para cambiar el brillo y la intensidad de la imagen, además, puede difuminar un poco la imagen. Esto ayuda a que el usuario no se distraiga con la imagen de fondo y se centre en el contenido de la web.

Otra cosa a tener en cuenta sobre las imágenes es que si quiere añadir una imagen a su web y dicha imagen es más pequeña que el hueco donde la quiere añadir y tiene un color de fondo distinto al de su web, tendrá que ser conscientes de que se va a ver el fondo de la propia imagen y el fondo de la web. Esto producirá un efecto visual poco atractivo, ya que, tendrá el color de fondo de la web y otro color de fondo rodeando a la imagen. Para corregir esto, siempre que la imagen sea más pequeña que el hueco donde va a añadirla, debe definir el fondo de la imagen como color transparente. Así, una vez insertada la imagen, solo se verá el color de fondo de su web y evita el problema.

Para finalizar, debe tener en cuenta que las imágenes que estén incluidas en su web, se transferirán al usuario cuando este entre a la web a través de la red, ya sea local en el propio servidor o a través de Internet. Por lo que, si la web tiene muchas imágenes y, además, son imágenes muy pesadas, el tiempo de carga de la página aumentará y, si el usuario tiene una conexión de red mala, puede que no pueda visualizar bien el contenido.

### **2.1.2. Optimización de imágenes**

Es importante que las imágenes que utilice tengan un tamaño adecuado y esto se debe a tres factores principales:

- 1. La navegación móvil.** Como ya sabe el uso del móvil es cada vez mayor y la velocidad que proporcionan las redes 4G y 5G no puede asemejarse con la banda ancha que aporta la fibra óptica. Por lo tanto, los usuarios de dispositivos móviles agradecerán que su web no tarde en cargar mucho rato.
- 2. Pérdida de usuarios.** Todos saben que una web lenta es sinónimo de perder usuarios, lo que se traduce como un porcentaje de rebote muy alto. Para que esto no suceda, es importante que optimice su web y este proceso empieza por optimizar el peso de sus imágenes.
- 3. Posicionamiento SEO.** Los distintos buscadores disponen de un tiempo limitado para rastrear las webs, es decir, que cuanto menos pese su web, más páginas podrá rastrear y tendrá más posibilidades de posicionar mejor su web. Además, una de las cosas en las que Google hace mucho hincapié actualmente es la velocidad de carga. Por lo que, si mejora la velocidad de carga gana puntos y su posicionamiento mejorará.

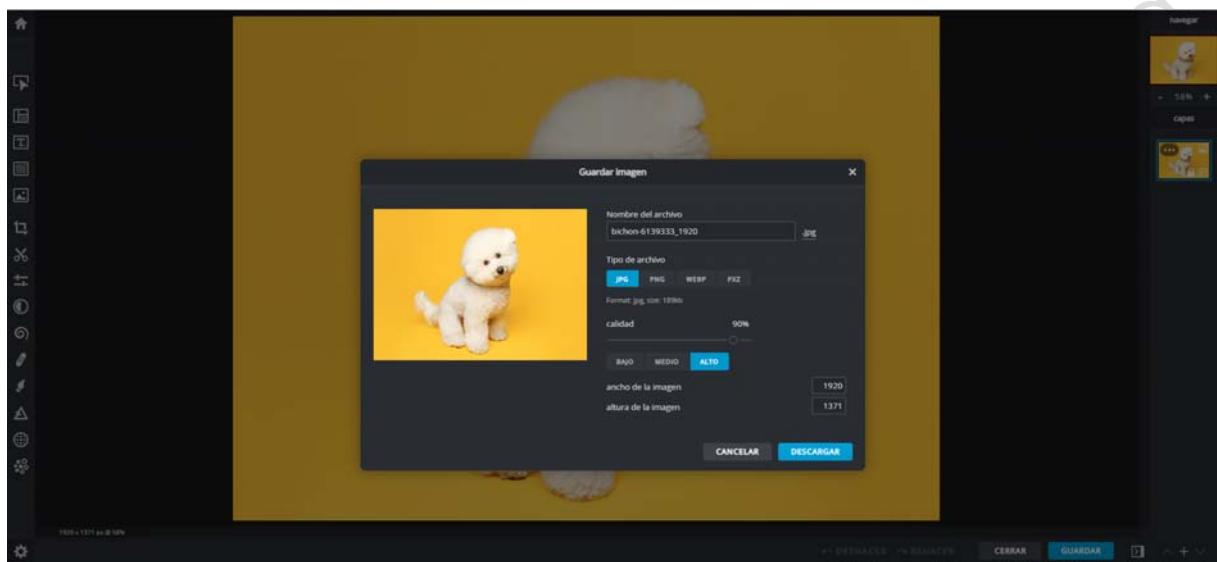
Como ya ha visto anteriormente, las imágenes para una web deben estar en formato GIF, en PNG o en JPG y los archivos deberán tener sus respectivas extensiones. Este punto es importante porque de este modo es cómo el navegador es capaz de reconocer el tipo de archivo que está procesando.

Imagen .gif. Hoy por hoy este tipo de imágenes están obsoletas. Solo se utilizan cuando quiera añadir una imagen animada.

Imagen .png. Como se ha visto anteriormente, estas imágenes sustituyen a las imágenes gif, puesto que también permiten utilizar el color transparente. Este formato es el mejor a utilizar para imágenes planas o con mucho espacio en blanco como, por ejemplo, logos, capturas de pantalla, productos para tienda online, etc. Es cierto que el formato png no comprime igual de bien como jpg, en cambio aporta imágenes con mayor calidad.

Imagen .jpg. Este formato es muy útil para utilizarlo con fotografías con muchos colores y detalles, ya que, es un formato capaz de comprimir al máximo las imágenes, aunque la desventaja que tiene es que la imagen también perderá calidad y nitidez.

Uno de los mejores programas para optimizar imágenes es Photoshop. Si no tiene Photoshop, también puede utilizar diferentes herramientas online que permiten optimizar sus imágenes sin necesidad de instalar una aplicación. Para este ejemplo, se usa el editor de imágenes online [Pixlr](#). Vea el ejemplo a continuación:



Como puede observar en la imagen, puede cambiar la calidad de la propia imagen para que pese menos. De hecho, si le aplica una compresión de calidad media, baja de 159 kb que pesa la imagen original a 108 kb. Aunque parezca poco la compresión realizada, hay que pensar que en una web puede haber más de 100 imágenes, por lo tanto, si suma todo lo que pesa cada imagen la compresión es considerable.

Por otro lado, para ayudar al SEO de la imagen debe poner un título que la describa en sí. Por ejemplo, en esta imagen podría poner un título como el de perro-bichon-maltes y, esto ayudaría mucho al SEO y por ende a posicionar la web. Al ponerle nombre a sus imágenes hay que considerar que debe evitar los espacios en blanco y las tildes.

Al poner la imagen también debe tener en cuenta rellenar el atributo alt con una breve descripción de la imagen. Esto sirve para por si en algún caso no se pudiera cargar la imagen, los usuarios puedan ver la descripción de la misma, en vez de un ícono de imagen rota. En este atributo debe añadir palabras a las que llama keywords. Estas palabras son palabras claves que determinará analizando qué palabras utilizarían los usuarios para buscar lo que les está ofreciendo. Además, también puede ayudar con la herramienta de Google Analytics para saber cuáles son las palabras que más buscan los usuarios en la web. En su imagen de ejemplo pondrá como keywords alt=Perro raza bichon maltes. Hay que tener especial cuidado en utilizar keywords en imágenes decorativas tipo fondo de pantalla, botones, bordes, etc. Pues Google las penaliza.



TOME NOTA

*Se deben utilizar las mismas keywords en diferentes imágenes de un mismo producto. Por ejemplo, si en una tienda online tiene 3 fotografías de un mismo producto, debe etiquetarlas del mismo modo.*

### 2.1.3. Implementación de código con varias imágenes

Como se ha visto anteriormente, las imágenes no solo sirven como decoración para un documento o para aportarle información adicional, las imágenes también pueden funcionar como hiperenlaces. Las imágenes que funcionan como hiperenlaces se pueden ver muy a menudo, por ejemplo, en los blogs. Crear una imagen que funciona como hipervínculo es tan sencillo como introducir la etiqueta img como contenido del elemento. Vea el ejemplo de cómo codificar imágenes como hiperenlaces:

```



```

Al añadir una imagen como hipervínculo, lo único que tendrá de diferencia al resto de imágenes es que aparecerá un pequeño borde azul rodeando la imagen. Esto se puede cambiar utilizando código CSS. Otra diferencia que puede aparecer en algunos navegadores cuando tiene imágenes como hiperenlace, es que el puntero del ratón cambie de forma al pasar sobre ellas. De todas maneras, la imagen debe darle indicios al usuario de que se trata de un enlace con el que puede interactuar, ya que, si no es así los usuarios no interactuarán con la imagen y por lo tanto no cumplirá su cometido.

Además, las imágenes no solo pueden ser hiperenlaces, sino que también pueden ser mapas de imágenes donde determinadas áreas están programadas para que el usuario pueda interactuar con ella. Hay dos tipos de mapas de imagen, las que están basadas en el servidor y las que están basadas en el cliente. Un mapa de imagen basado en el servidor se crea con el atributo ismap de la etiqueta img. Se caracteriza porque cuando desplaza el puntero del ratón por encima de la imagen y hace clic, es capaz de capturar las coordenadas, expresadas en píxeles que comienzan desde la esquina superior izquierda de la imagen, y enviar como parámetro estos datos a través de la url que indique la etiqueta href. Luego, en el servidor habrá un programa que tomando estos datos sea capaz de procesar una respuesta. Vea cómo se comporta un mapa de imagen observando el siguiente código:

```


```

</a>

Por otro lado, los mapas de imagen basados en el cliente están compuestos por una etiqueta img con el atributo usemap. Este atributo apunta a una lista de coordenadas, formas e hiperenlaces. El mapa está generado mediante los elementos área y map. En este ejemplo, es el propio navegador quien decide qué url debe utilizar según donde se haya clicado. Esto se llevará a cabo guiándose por las coordenadas establecidas. Este tipo de mapas de imagen tiene una gran ventaja en comparación con los mapas de imagen basados en el servidor y, es que no precisa ningún programa para interpretar los mapas de imagen que se ejecute en el servidor.

Para utilizar los mapas de imagen, primero, debe subir la imagen o imágenes al servidor. Después, tan solo tendrá que coger las url donde están almacenadas las imágenes. Las url deberán tener el formato que se observa a continuación:

`http://nombreServidor/images/nombreImagen1.jpg`

`http://nombreServidor/images/nombreImagen2.jpg`

Cuando tenga las url de las imágenes, se creará un módulo en HTML y se añadirá el código que se presenta a continuación:

```

```

En la parte del código donde pone urlImagen es donde debe insertar la url de cada imagen que quiera añadir. Por cada imagen, debe escribir un nuevo código HTML. Por ejemplo, si quiere añadir dos imágenes quedaría así:

```



```

Con este código conseguirá tener dos imágenes en línea.

También, puede crear una tabla, por ejemplo, de tres filas y dos columnas e insertar una imagen en cada una de las celdas. Vea cómo sería esta disposición de imágenes:

Img 1	Img2
Img3	Img4



A continuación, vea el código HTML que permitirá crear esta estructura:

```
<html>

<head>

<title>Imágenes en tabla</title>

<meta charset="utf-8">

</head>

<body>

<table width="80%" border="0">

<tr align="center" valign="middle">

<td></td>

<td></td>

</tr>

<tr align="center" valign="middle">

<td></td>

<td></td>

</tr>

<tr align="center" valign="middle">

<td></td>

<td></td>

</tr>

</table>

</body>

</html>
```

El **primer paso** para crear un mapa de imagen basado en el cliente es definir un mapa de coordenadas. Como ha visto anteriormente, el mapa de imagen basado en el cliente debe contener un atributo map que como es obvio representa al mapa, y dentro de map, deberá haber una o más áreas configuradas. El conjunto de estos elementos establecería las url, las coordenadas y los atributos que debe albergar cada área de la imagen que vaya a definirse como hipervínculo. Para utilizar map, debe emplear una etiqueta de apertura y otra de cierre de manera obligatoria, es decir, <map> y </map>. En el interior de estas dos etiquetas puede introducir los atributos del bloque y todas las áreas que necesite para crear su mapa de imagen.

Añadirá un elemento área por cada zona del mapa de imagen que quiera diferenciar. Al contrario de lo que pasaba con el elemento map, en el elemento área debe utilizar una etiqueta de apertura, pero puede optar por poner o no la etiqueta de cierre. Lo más importante del elemento área son sus atributos y, en especial href, coords o shape. Con el atributo coords, indicará las coordenadas; con el atributo href, la url donde se redigirá según la zona donde la haya configurado y por último, el atributo shape indicará la forma geométrica del propio área. Además, el atributo shape puede tomar tres valores diferentes. Los valores son poly, circle y rect. Estos valores van depender de si el área a definir es un polígono, un círculo o un rectángulo. Dependiendo cuál de los tres valores tome el atributo shape, dependerán los datos del atributo coords.

Puede crear todos los mapas de coordenadas que necesite. Una vez haya terminado de definir el mapa o los mapas de coordenadas, debe asociar ese mapa con la imagen que vaya a funcionar como mapa de imagen. En este punto es donde se utiliza el atributo usemap, se le asigna el nombre del mapa estableciéndolo mediante el atributo name de la etiqueta map. Para insertar el nombre, debe usar la almohadilla y, después, escribir el nombre. Vea el ejemplo: #nombreMapa. Opcionalmente, puede utilizar el atributo longdesc donde puede insertar una url que transporte a una página donde haya una descripción del mapa de imagen. Esta opción es muy útil para hacer que su página sea accesible para invidentes. Por otra parte, a usemap, le añadirá una url la cual dirija a un documento donde se haya definido el mapa de coordenadas para la imagen, aunque cabe decir que a usemap le podría insertar cualquier otra url.

Al igual que cada persona tiene una caligrafía diferente donde unas personas tienden a escribir más ordenadamente, otras a amontonar las letras, también a darle inclinación a la caligrafía, etc. Un ordenador también es capaz de utilizar diferentes tipos de caligrafías. A estas caligrafías se le suele denominar fuentes. Cada sistema operativo o aplicación dispone de determinados tipos de fuentes, aunque, posteriormente también puede añadir diferentes tipos de fuentes. Puede encontrar fuentes de uso gratuito y otras de pago. Utilizando el atributo basefont o Font, puede elegir cuál de las diferentes opciones de fuentes disponibles quiera usar en el texto de su documento HTML. Para ello, debe saber cómo se llama la fuente que gusta y añadir el nombre al atributo. Además, en este atributo puede indicar más de un tipo de fuente determinando un orden de prioridad, por si un usuario abre el documento en un navegador que no acepta el tipo de fuente indicado como primero, tenga la posibilidad de poder usar el segundo tipo que ha establecido. Las fuentes en sí, plantean algunos problemas de compatibilidad, ya que, las fuentes no dependen del estándar de HTML, ni de los navegadores, sino de los propios dispositivos. Por lo tanto, cada sistema tiene sus propias fuentes y, en general, muchas veces las fuentes tienen derechos de autor y su distribución queda limitada. Hay un conjunto amplio de fuentes que son aceptadas por todos los sistemas, sin embargo, también

encuentra algunos tipos de fuentes que son específicas al propio sistema. Es por ello, que siempre debe establecer más de un tipo de fuente en su documento HTML.

Por lo general, cuando escoge un tipo de tipografía en su página, lo que quiere es diferenciar las distintas partes del documento. Sin embargo, si el navegador no es capaz de leer la fuente que ha escogido, sustituirá el texto por una fuente por defecto y las partes de su documento quedarán sin distinción alguna. Para evitar este problema, existen tipografías genéricas como ya se ha comentado anteriormente. Algunas de las tipografías genéricas son: sans-serif, serif, monospace, fantasy y cursive. Estos cinco tipos de fuentes, no son fuentes reales, tan solo son indicaciones que le da al navegador para que utilice una fuente sin adornos, con adornos, con espacio fijo, de fantasía o cursiva respectivamente. El propio navegador asociará estas indicaciones con una fuente queG haya disponible y cumpla con estos requisitos. Por otro lado, si está diseñando una web de solo uso interno de una empresa, sí que puede especificar una fuente real, ya que, todos los usuarios tendrán los mismos sistemas y no habrá problema de compatibilidad.

Además, también puede utilizar fuentes reales combinadas con fuentes genéricas para que así siempre obtenga como resultado un cambio de fuente. Para hacer eso, en primer lugar, se establece la fuente que realmente quiera insertar y en segundo lugar la genérica. Así, si hay algún sistema que no interprete la primera, escogerá la segunda y cambiará la letra por defecto. Vea el ejemplo:

```

```

Esta línea de código describe que, si en el sistema se encuentra la fuente Helvetica, la página utilice esta. En cambio, si la fuente Helvetica no está disponible, pasará a utilizar la Arial y, en caso de que tampoco se encuentre la fuente Arial, pasará a utilizar la fuente por defecto Sans-Serif, que es la clasificación a la que pertenecen las dos anteriores.

Ya se ha hablado de las fuentes por defecto de los sistemas, pero también existe un tamaño por defecto predefinido por los sistemas. Normalmente, el tamaño de la letra puede cambiar porque el usuario configure su sistema para que muestre la letra más grande o más pequeña, esto cambia dependiendo del gusto y de la visión de cada usuario. Sin embargo, en su documento HTML también puede configurar el tamaño de la letra. Para ello, se utiliza el atributo size de Font y de fontbase donde se estipula un valor para el tamaño de la letra, aunque este valor no va a estar expresado en ninguna unidad real como píxeles o puntos, ya que el tamaño será relativo al usuario, como bien se ha comentado anteriormente. Por lo tanto, el valor que puede asignar al atributo size, va a estar comprendido entre 1 y 7. Así, es cómo elegir uno de los 7 tamaños predefinidos por el sistema. Además, cabe destacar que también puede utilizar un signo + o – para cambiar el tamaño relativo del texto. Sin embargo, si, por ejemplo, quiere cambiar el tamaño de una parte del texto, es recomendable utilizar los atributos small y big, en vez de utilizar Font.

Por otra parte, para cambiar el color de la fuente debe establecerlo en los elementos Font y fontbase, aunque esto se realiza cuando solo quiere cambiar el color de una parte del texto. En cambio, puede establecer el color por defecto de todo el texto en la etiqueta body. Para establecer el color del texto, si es un color básico puede expresarlo con su nombre en inglés, en cambio si es otro

tipo de color puede usar su código hexadecimal. Vea un ejemplo donde puede observar el color verde expresado con su nombre en inglés y el color blanco expresado en hexadecimal:

```
Texto
```

```
Texto
```

La primera línea de código mostrará el texto tamaño 4 de color verde. La segunda línea mostrará el texto tamaño 4 en color blanco.

Cuando utiliza en un documento diversos colores, lo hace pensando en visualizar el documento en un formato o dispositivo capaz de interpretar esos colores. El único problema es que esos colores pueden cambiar un poco su tonalidad dependiendo de la calibración del dispositivo. Es por ello que hay que tener en cuenta dónde se van a visualizar estos colores, ya que, para formatos a papel el código de colores que va a interpretar la impresora es CMYK que equivale a Celeste, Magenta, Amarillo y Negro. Si mira un cartel con una lupa, verá que los diferentes colores se constituyen con pequeños puntos celestes, magenta y amarillo, siendo el negro la mezcla de los tres colores base. Por otro lado, cuando utiliza los colores en un dispositivo digital, el código de colores que interpretan estos dispositivos es el RGB, es decir, los colores Rojo, Verde y Azul, donde el blanco es la combinación de los tres.

Puesto que la mayoría de documentos programados en HTML van a ser visualizados en dispositivos digitales, el código de colores que utilizar para definir los colores en su documento HTML va a ser el código RGB. Aunque, se debe tener en cuenta que hay que ser muy consciente de qué y cómo va a utilizar estos colores, ya que, si por ejemplo cualquier tabla o elemento de la web está pensada para ser impresa en un futuro, debe adecuar los colores para que no disten del concepto original.

Los desarrolladores web y diseñadores gráficos suelen utilizar sistemas que les permiten adaptar los colores y así, consiguen, trabajar con millones de tonos de color. También, utilizan las conocidas pantallas de retina que muestran un color lo más real posible. Además, deben calibrar sus pantallas también para el mismo cometido. En cambio, aunque estos profesionales se esfuerzan al máximo para adecuar los colores y que sean lo más reales posibles, si el usuario utiliza dispositivos más limitados, no va a ver el mismo tono de color que se le muestra realmente, ya que, dependiendo los dispositivos son capaces de traducir más o menos tonos de color. Por ejemplo, los ordenadores antiguos solo eran capaces de mostrar 16 colores.

Para finalizar, va a ver los colores básicos que se definen tanto en HTML 5 como en CSS, a los cuales puede acceder mediante sus nombres, como ya se ha visto en un ejemplo anterior. Los denomina colores básicos porque estos colores existen en todos los dispositivos, así que, estos colores son los que verá en su forma real un usuario a través de un dispositivo normal. Vea cuáles son:

- Navy: Azul oscuro
  - Yellow: Amarillo
  - Olive: Verde oscuro
  - Lime: Verde lima
-

- Green: Verde
- Fuchsia: Fucsia
- Purple: Morado
- Red: Rojo
- Maroon: Granate
- White: Blanco
- Gray: Gris
- Silver: Plata
- Black: Negro
- Teal: Azul cerceta
- Aqua: Celeste
- Orange: Naranja
- Darkred: Rojo oscuro



TOME NOTA

*Hay muchos más colores básicos que puede encontrar en el siguiente [enlace](#). Estos colores son colores predefinidos que puede nombrar fácilmente al utilizar código HTML o CSS..*

Tanto en HTML como CSS, todos los colores predefinidos y no predefinidos se componen con valores RGB. Los valores RGB se componen de tres bytes, es decir, con un total de 24 bits. Cada byte corresponde a uno de los componentes, es decir, un byte para el rojo, un byte para el verde y un byte para el azul. Con un byte, o lo que es lo mismo, 8 bits se pueden representar un total de 256 valores diferentes. Estos valores pueden representarse de 0 a 255 en decimal o de 0 a FF en hexadecimal. Por lo tanto, todo esto significa que para el color rojo habrá 256 tonos de rojo, para el verde 256 tonos de verde y para el azul 256 tonos de azul. En definitiva, si calcula las combinaciones de colores sale que por cada tono de rojo tiene 256 de verde y 256 de azul. Por lo tanto, tendría 256x256x256, lo que equivaldría a 16777216 colores diferentes. Como es obvio, muchas tonalidades del mismo color no son perceptibles para el ojo humano, por lo tanto, aunque un dispositivo pueda mostrar más de 16 millones de colores, muchos de ellos van a parecer el mismo tono de color.

Para utilizar el sistema hexadecimal para insertar un color, tan solo debe utilizar el atributo color del elemento Font. También hay otros tipos de atributos donde puede insertar el color. Vea un ejemplo donde se inserta un color en hexadecimal:

Color="#ffaacc"

Siempre que vaya a insertar un color en hexadecimal debe utilizar el símbolo de la almohadilla. Este símbolo indica que el código utilizado es RGB en hexadecimal. El valor ff equivale a la parte del rojo, el valor aa a la parte del verde y el valor cc a la del azul. Estos tres colores deben tener un valor comprendido entre 0 y F.

## 2.2. Trabajar con textos: efectos estéticos y de movimiento

### Efectos estéticos

Al crear documentos HTML para ser visualizados de manera digital o en papel, debe tener en cuenta que tiene que utilizar bien los colores para elaborar el propio documento, ya que, que utilice bien o mal los colores equivaldrá al mayor o menor atractivo de este. Dependiendo de los colores que utilice para su web le da un estilo más formal, alegre, juvenil, etc.

Como vio anteriormente, cuando crea una página web suele utilizar tipos de letras diferentes para delimitar las diferentes zonas de la propia web. Al decir, tipo de letra está refiriendo no solo a que la letra esté en negrita, subrayada o cursiva, sino a la propia tipografía de la letra. Además, en una web lo lógico es escribir lo más importante con un tamaño de letra mayor y lo que resulte menos importante lo escribe con un tamaño de letra menor. Así, le da una jerarquía a la información de la página. También, es importante la elección del color del texto, ya que, como ha citado anteriormente, el color puede dar una impresión favorable o desfavorable de lo que está leyendo.

Para realizar estas diversas configuraciones sobre el texto de la página, utiliza los elementos Font y fontbase, aunque estos elementos están obsoletos según W3C. Pero, antes de aprender la nueva forma de codificar las opciones del texto, va a ver cómo configurarlas con los elementos Font y fontbase y después aprenda la manera más adecuada de hacerlo. Debe recordar que la mejor manera de codificar los estilos de su web utilizando código CSS.

Al utilizar el elemento fontbase se establecen las propiedades de la tipografía que se utilizará en todo el documento. Esto sucederá desde el punto del documento que aparece el elemento fontbase hacia abajo. Recuerde que el flujo del programa va siempre desde arriba hacia abajo del documento.



*Debe tener en cuenta que hay navegadores como Safari o Mozilla Firefox que no son capaces de procesar el elemento fontbase. Sin embargo, el explorador de Internet Explorer sí es capaz de traducir las instrucciones que establezca en el elemento fontbase.*

TOME NOTA

Desde un principio, el código HTML fue ideado para trabajar con texto, por lo que los elementos que permiten trabajar con imágenes en HTML fueron introducidos posteriormente como un extra. Además, HTML desde sus comienzos siempre ha tenido un carácter atractivo para ser utilizado como transmisor de la información en los motores de búsqueda, ya que, es un código fácil de utilizar y muy flexible. Por lo que, las empresas que crearon los propios navegadores han ayudado también al desarrollo de HTML, facilitando así la labor de los diseñadores web.

Para darle formato al texto de su web con HTML, tan solo debe rodear al texto deseado con las etiquetas correspondientes para cada recurso requerido. Por ejemplo, si quiere emplear un texto como título principal de una web, debe emplear las etiquetas H1 de la siguiente manera:

<H1>Título de su web</H1>

Esto le indica al navegador, que muestre el texto del interior de las etiquetas en letra grande y que, además, ese texto es de mayor importancia que el resto. Esto es muy importante, puesto que, diferenciar los diferentes tipos de texto en su documento ayuda al SEO de su web y por lo tanto al posicionamiento de la misma. De la misma manera que ha establecido el título principal, puede establecer hasta 6 tipos de títulos, siendo el h1 el de mayor interés e importancia y el h6 el de menor. Es importante dominar esta jerarquía de la información. Vea un ejemplo donde se usan las diferentes etiquetas HTML:

```
<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8">

<title>Ejemplo de código HTML</title>

</head>

<body>

<h1>Título de mayor importancia y mayor tamaño de letra</h1>

<h6>Título de menor importancia y menor tamaño de letra</h6>

</body>

</html>
```

Vea a continuación el resultado de cómo muestra el navegador este código:

## Título de mayor importancia y mayor tamaño de letra

Título de menor importancia y menor tamaño de letra

Por otro lado, tiene la etiqueta Font que permite escoger el tipo de fuente con la que desea que se muestre el texto de su navegador. Como es obvio, la fuente que elija deberá estar instalada previamente en su sistema operativo. Vea un ejemplo:

```
<!DOCTYPE html>
```

```
<html>
<head>
<meta charset="utf-8">
<title>Ejemplo de cambio de fuente</title>
</head>
<body>
Texto en Verdana

Texto en Impact

</body>
</html>
```

Vea el resultado que muestra el navegador a continuación:

Texto en Verdana  
**Texto en Helvetica**

A continuación, va ver el uso de la etiqueta párrafo. Esta etiqueta es una etiqueta única, es por ello, que puede utilizar la etiqueta de cierre, pero no es algo obligatorio. Vea a continuación un ejemplo:

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>Ejemplo con etiqueta de párrafo</title>
</head>
<body>
<p>Esto es un párrafo</p>
<p>Aquí, empieza el siguiente párrafo</p>
```

```
</body>
```

```
</html>
```

El resultado del código que ha escrito será el siguiente:

Esto es un párrafo

Aquí, empieza el siguiente párrafo

El próximo elemento que verá es el atributo align de la etiqueta párrafo. Este elemento permitirá cambiar la posición de alineación del párrafo. Para ello, utiliza los parámetros right, left y center para alinear el párrafo a la derecha, izquierda y centro respectivamente. Vea un ejemplo:

```
<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8">

<title>Alineación de párrafo</title>

</head>

<body>

<p align="right">Esto es un párrafo alineado a la derecha</p>

<p align="left">Esto es un párrafo alineado a la izquierda</p>

<p align="center">Esto es un párrafo alineado al centro</p>

</body>

</html>
```

El resultado de este código en el navegador es el siguiente:

---

Esto es un párrafo alineado a la izquierda

Esto es un párrafo alineado al centro

Esto es un párrafo alineado a la derecha

---

Las últimas etiquetas que verá son:

<br/> que se utiliza para saltar una línea.

<b>Texto</b> La utiliza para poner un texto en negrita.

<strong></strong> También sirve para poner un texto en negrita.

<i>Texto en cursiva</i> Se utiliza para poner un texto en cursiva.

<em>Texto</em> También se utiliza para poner un texto en cursiva.

<u>Texto</u> Se utiliza para subrayar un texto.

Estas etiquetas son las que puede utilizar en HTML, aunque actualmente se encuentran en desuso. En cambio, lo que se utiliza actualmente para crear este tipo de efectos y poder modificar los textos de su web es el código CSS. Con el código CSS será capaz de formatear cualquier texto como apetezca, evitando los textos por defecto de los navegadores.

## Efectos de movimiento

Para realizar animaciones, utiliza una técnica denominada DHTML que utiliza JavaScript y CSS. Crear animaciones con DHTML no tiene nada especialmente particular, lo único que debe hacer es coger un elemento y repetir determinadas veces un movimiento para que este termine en la posición deseada. Imagine que quiere coger una imagen que se desplace hacia abajo, pues solo tendrá que incrementar ese desplazamiento las veces necesarias para que la imagen llegue a la posición deseada. También podría modificar de manera continua la propiedad style.clip para mover la imagen píxel a píxel.

### 2.2.1. Creación de textos mejorados y de movimiento

Para crear textos mejorados, va a utilizar código CSS. Antes que nada, va a observar las diferentes propiedades que puede utilizar para realizar textos mejorados. Vea la siguiente tabla:

Propiedad	Descripción	Valor	Detalles
<b>Text-indent</b>	Desplaza la primera línea de texto	Longitud	Longitud
<b>Text-align</b>	Alineamiento de texto	Left Right Center	Izquierda Derecha Centro

		justify	Justificado
<b>Text-decoration</b>	Efectos del texto	None Underline Overline Line-through Blink	Ninguno Subrayado Línea por encima Tachado Parpadeo
<b>Text-transform</b>	Transformaciones de texto	Capitalize Uppercase lowercase	Pone en mayúscula la primera letra de cada palabra.  Pone todas las letras en mayúsculas  Pone en minúsculas todas las letras
<b>Letter-spacing</b>	Espacio entre caracteres	None Normal longitud	Ninguno Normal Longitud
<b>Word-spacing</b>	Espacio entre palabras	Normal longitud	Normal Longitud
<b>White-space</b>	Comportamiento de espacios dentro de los elementos	Pre Nowrap Pre-wrap Pre-line	Preformatado  Cambios de línea solo al utilizar br  Pre-line
<b>Direction</b>	Sentido de la escritura	Ltr Rtl	Izquierda a derecha  Derecha a izquierda
<b>Unicode-bidi</b>	Sentido de la escritura	Normal Embed Bidi-override	Normal  Incrusta algoritmo bidireccional  Crea una sustitución si el elemento es a nivel de línea o a nivel de bloque

A continuación, vea algunos ejemplos de lo que ha visto:

**Atributo text-indent**, recuerde que para utilizar este atributo debe indicar una longitud. Sangría de párrafo de 3cm. Ejemplo:

```
<!DOCTYPE html>

<html>

<head>

<style type="text/css">p{text-indent:3cm}</style>

<meta charset="utf-8">

<title>Ejemplo de código HTML</title>

</head>

<body>

<p>Esto es un párrafo con una sangría de 3cm con respecto al borde</p>

</body>

</html>
```

Vea el resultado de este código en el navegador:

Esto es un párrafo con una sangría de 3cm con respecto al borde

**Atributo text-align**, recuerde que para utilizar este atributo debe utilizar left, right, center o justify. Alineación de los textos a la derecha, a la izquierda y al centro. Ejemplo:

```
<!DOCTYPE html>

<html>

<head>

<style type="text/css">

p.derecha{text-align:right}

p.izquierda{text-align:left}
```

```
p.centro{text-align:center}

/>

<meta charset="utf-8">

<title>Ejemplo de código HTML</title>

</head>

<body>

<p class="derecha">Esto es un párrafo alineado a la derecha</p>

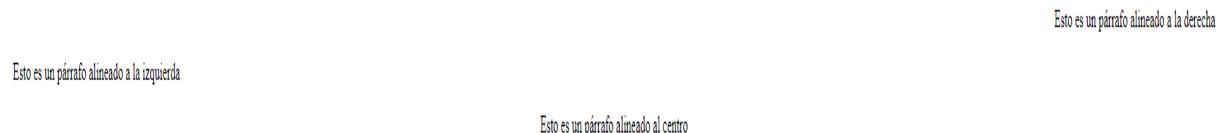
<p class="izquierda">Esto es un párrafo alineado a la izquierda</p>

<p class="centro">Esto es un párrafo alineado al centro</p>

</body>

</html>
```

Resultado que muestra el navegador:



Esto es un párrafo alineado a la izquierda  
Esto es un párrafo alineado al centro  
Esto es un párrafo alineado a la derecha

**Atributo letter-spacing**, para utilizar este atributo utiliza un valor numérico o pondrá normal. Separación entre letras de 8 píxeles. Ejemplo:

```
<!DOCTYPE html>

<html>

<head>

<style type="text/css">

p{letter-spacing:8px}

</style>

<meta charset="utf-8">
```

```
<title>Ejemplo de código HTML</title>

</head>

<body>

<p>Distancia entre letras de 8 píxeles</p>

</body>

</html>
```

Resultado que muestra el navegador:

D i s t a n c i a   e n t r e   l e t r a s   d e   8   p í x e l e s

**Atributo text-transform**, recuerde que para utilizar este atributo debe señalar las opciones de capitalize, uppercase, lowercase y none. Vea un ejemplo:

```
<!DOCTYPE html>

<html>

<head>

<style type="text/css">

p.cap{text-transform:capitalize}

p.uCase{text-transform:uppercase}

p.lCase{text-transform:lowercase}

p.non{text-transform:none}

</style>

<meta charset="utf-8">

<title>Ejemplo de código HTML</title>

</head>

<body>
```

```
<p class="cap">Párrafo con opción capitalize</p>
<p class="uCase">Párrafo con opción uppercase</p>
<p class="lCase">Párrafo con opción lowercase</p>
<p class="non">Párrafo con opción none</p>
</body>
</html>
```

Resultado que muestra el navegador:

Párrafo Con Opción Capitalize  
PÁRRAFO CON OPCIÓN UPPERCASE  
párrafo con opción lowercase  
Párrafo con opción none

**Atributo text-decoration**, recuerde que las opciones que puede utilizar en este atributo con none, underline, overline, line-through y blink. Vea el ejemplo:

```
<!DOCTYPE html>

<html>

<head>

<style type="text/css">

p.none{text-decoration:none}

p.uLine{text-decoration:underline}

p.oLine{text-decoration:overline}

p.lThrough{text-decoration:line-through}

p.blink{text-decoration:blink}

</style>

<meta charset="utf-8">
```

```
<title>Ejemplo de código HTML</title>

</head>

<body>

<p class="none">Párrafo con opción none</p>

<p class="uLine">Párrafo con opción underline</p>

<p class="oLine">Párrafo con opción overline</p>

<p class="lThrough">Párrafo con opción line-through</p>

<p class="blink">Párrafo con opción blink</p>

</body>

</html>
```

Resultado que muestra el navegador:

Párrafo con opción none  
Párrafo con opción underline  
Párrafo con opción overline  
~~Párrafo con opción line-through~~  
Párrafo con opción blink

Atributo Word-spacing, recuerde que este atributo tiene las opciones: normal y una distancia determinada por un valor que elija. Vea el ejemplo:

```
<!DOCTYPE html>

<html>

<head>

<style type="text/css">

p{word-spacing: 2cm;}

</style>
```

```
<meta charset="utf-8">

<title>Ejemplo de código HTML</title>

</head>

<body>

<p>Las palabras de este párrafo tienen una distancia de 2cm entre ellas</p>

</body>

</html>
```

Resultado que muestra el navegador:

Las palabras de este párrafo tienen una distancia de 2cm entre ellas

Atributo White-space, recuerde que este atributo puede utilizar las opciones pre, nowrap, pre-wrap, normal y pre-line. Vea un ejemplo:

```
<!DOCTYPE html>

<html>

<head>

<style type="text/css">

p.normal{white-space: normal;}

p.pre{white-space: pre;}

p.nowrap{white-space: nowrap;}

p.preWrap{white-space: pre-wrap;}

p.preLine{white-space: pre-line;}

</style>

<meta charset="utf-8">

<title>Ejemplo de código HTML</title>

</head>
```

```
<body>

<p class="normal">Párrafo con opción normal</p>

<p class="pre">Párrafo con opción pre</p>

<p class="nowrap">Párrafo con opción nowrap</p>

<p class="preWrap">Párrafo con opción pre-wrap</p>

<p class="preLine">Párrafo con opción pre-line</p>

</body>

</html>
```

Resultados que muestra el navegador.

Párrafo con opción normal

Párrafo con opción pre

Párrafo con opción nowrap

Párrafo con opción pre-wrap

Párrafo con opción pre-line



**Debe tener en cuenta que al utilizar la opción blink del atributo text-decoration, no va a funcionar en todos los navegadores. Por ejemplo, en Internet Explorer no va a funcionar. Sin embargo, en Mozilla Firefox sí.**

TOME NOTA

Si domina estas propiedades, será capaz de crear páginas donde los párrafos tendrán su propio estilo de espaciado, sangría, espaciado entre caracteres, distintas alineaciones, e incluso el parpadeo si utiliza la propiedad blink de text-decoration, aunque precisamente esta propiedad no servirá si lo que quiere es imprimir el contenido en papel.

Va a modificar el body de un documento para establecer la tipografía, el tamaño y el color de la letra para todo el documento. También, establece el color de fondo del documento, la alineación y el sangrado que aplica a cada párrafo del texto. Para que el documento se adapte a cualquier ancho de pantalla, aplica el atributo text-index con porcentaje. Esto hará que el sangrado del párrafo pueda ajustarse a todo tipo de pantallas, ya sean, de ordenador, de móviles o de tablets. Vea el ejemplo:

```
<body style="Font-family: Verdana, arial, sans-serif;
Font-size: 12pt; text-align: justify;

Text-indent: 10%; background: gold; color: black">
```

Además, para modificar líneas de texto específicas, tiene las etiquetas **<b>**, **<i>** y **<s>**. Debe tener en cuenta que algunas de estas etiquetas están obsoletas y que, además, son etiquetas demasiado específicas, ya que, como ha comentado anteriormente solo sirven para cambiar el texto que hay dentro de ellas.

Actualmente el estándar HTML proporciona una serie de elementos que puede utilizar para asociar un estilo determinado al texto, pero siempre siendo independiente al dispositivo final que procese el propio documento. Vea algunos de los elementos:

- **EM:** El texto que introduzca entre estas etiquetas se leerá en una pantalla como texto en cursiva. Pero si este mismo texto es leído por un sintetizador de voz, puede que le ponga énfasis a esta sección de texto.
- **STRONG:** El texto que ponga entre estas etiquetas se mostrará en pantalla como texto en negrita. En el caso de que el texto sea leído por un sintetizador de voz, el énfasis será mayor que el texto contenido en la etiqueta EM.
- **CITE:** El texto que esté contenido entre las etiquetas CITE, se mostrará como un texto citado.
- **VAR, KBD, SAMP y CODE:** Estas etiquetas se utilizan para indicar que el texto mostrado pertenece a un argumento de un programa, un texto a introducir por el teclado, el resultado resultante de un programa y código de ordenador respectivamente. Los textos contenidos entre estas etiquetas no se mostrarán de manera especial en un navegador, pero sí que pueden mostrarse de manera especial en otro tipo de dispositivos. Es por ello que, aunque en pantalla a veces no se muestren los resultados, siempre es importante diferenciar el texto en su documento a base de código.

A continuación, va a ver un elemento denominado marquesina. Las marquesinas son ventanas donde puede ver un texto en desplazamiento. La etiqueta para utilizar marquesinas es **<marquee>Texto</marquee>**. Por defecto, al utilizar estas etiquetas, aparecerá el texto por el margen derecho y desaparecerá por el margen izquierdo.

Para crear una marquesina más personalizada, puede añadir más parámetro como, por ejemplo, anchura y altura utilizando las etiquetas width y height respectivamente. Los valores que les puede dar a estas etiquetas pueden ser un valor fijo en píxeles o un porcentaje de la pantalla. Actualmente, lo normal es trabajar con porcentajes, así, es como se crea un diseño responsive que permita adaptarse a cualquier tipo de dispositivo. Vea un ejemplo:

```
<marquee width="40%" height=70>Soy una marquesina con el 40% de la ventana de ancho y 70
píxeles de alto</marquee>
```

También, puede añadir un alineamiento de texto en la marquesina utilizando la etiqueta align. Las opciones de alineamiento de texto en esta etiqueta son: top, si quiere que se alinee hacia arriba; middle, si quiere que se alinee a la mitad y bottom, si quiere que se alinee hacia abajo. Por lo que quedaría un ejemplo como el siguiente:

```
<marquee width=40% height=70 align=middle>Soy una marquesina con el texto alineado en el medio</marquee>
```

A la marquesina, también puede añadirle un color de fondo con la etiqueta bgcolor. Vea un ejemplo:

```
<marquee bgcolor="green">Soy una marquesina con el color de fondo verde</marquee>
```

Para cambiar el comportamiento de la marquesina, utiliza la etiqueta behavior. Si utiliza la opción alternate en esta etiqueta, el texto se moverá de un lado a otro sin desaparecer. Vea el ejemplo:

```
<marquee behavior="alternate">Texto que no desaparece</marquee>
```

También, puede utilizar la opción scroll de la etiqueta behavior que permite que el texto aparezca de la misma manera que por defecto. Vea el ejemplo:

```
<marquee behavior="scroll"
```

Además, puede cambiar la dirección del texto con la etiqueta direction, donde puede elegir las opciones right, left, up y bottom en las que el texto se moverá hacia la derecha, hacia la izquierda, hacia arriba o hacia abajo, respectivamente. Vea un ejemplo.

```
<marquee direction="up">Soy una marquesina que se mueve hacia arriba</marquee>
```

Otra de las propiedades que puede utilizar es la propiedad scrollamount que indica cuántos píxeles avanzaría el texto. Por ejemplo, si quiere que el texto avance de 100 píxeles en 100 píxeles, lo configura de la siguiente manera:

```
<marquee scrollamount=100>El texto avanza de 100 en 100 píxeles</marquee>
```

De esta manera, el texto avanzará muy rápido.

En cambio, si quiere que el texto avance más lento, utiliza la etiqueta scrolldelay donde debe indicar cada cuántos milisegundos va a avanzar el texto, es decir, cuanto mayor sea el número indicado, más lento irá el texto. Va a poner un ejemplo donde el texto avance cada 1000 milisegundos, lo cual avanzará extremadamente lento. Vea el ejemplo.

```
<marquee scrolldelay=1000>Este texto avanzará muy lento</marquee>
```

Además, también puede establecer el número de veces que quiere que aparezca el texto. Debe saber que por defecto es infinito. Para ello, utiliza la etiqueta loop e indica un valor que serán las veces que quiere que aparezca el texto. Vea el ejemplo:

<marquee loop="2">Este texto solo aparecerá dos veces</marquee>

Puede establecer una separación entre el texto de la marquesina y el borde de la propia ventana de la marquesina. Para ello, utiliza las etiquetas hspace y vspace y establezca los píxeles de separación que quiera. Hspace sirve para la separación del margen izquierdo y derecho y vspace para la separación de los márgenes de arriba y debajo de la ventana. Vea un ejemplo:

<marquee vspace=20; hspace=15>Texto separa de los bordes de la ventana</marquee>

Por último, también puede cambiar la tipografía del texto de la marquesina aplicando las etiquetas Font con el atributo fase, todo ello fuera de la etiqueta marquee. Vea un ejemplo:

<Font fase="Verdana"><marquee>Texto con tipografía Verdana</marquee></Font>

### **2.2.2. Implementación de efectos**

Para implementar efectos en los textos, puede utilizar código HTML directamente en las etiquetas que permiten dar los efectos que quiere. Por ejemplo, si quiere poner una tipografía determinada a su texto y un color determinado lo hará como en el siguiente ejemplo.

<Font-family="Verdana, Arial" Font-color="black" Font-size="12">Texto con implementación de efectos</Font>

Sin embargo, lo ideal es emplear el código HTML junto con código CSS para realizar el formateo de su texto. Siempre que implemente código CSS dentro de HTML, lo debe introducir en la parte del head. Vea el siguiente ejemplo:

```
<!DOCTYPE html>

<html>

<head>

<style type="text/css">

.Esto1{

font-family:Verdana, Geneva, Tahoma, sans-serif;

font-size: 14px;

color:#ED1E71;

}

</style>
```

```
<meta charset="utf-8">

<title>Ejemplo de código HTML y CSS</title>

</head>

<body>

<div align="left">Texto con estilo establecido con CSS</div>

</body>

</html>
```

### **2.2.3. Adecuación de efectos a la página web**

Debe saber que cuando implementa código CSS en un documento HTML, el código CSS solo afectará a ese documento, es decir, no será reutilizable si en un futuro quiere crear un documento con los mismos estilos que ya tiene programados. Aunque, puede ir programando los mismos estilos en todos los documentos HTML, el gran inconveniente es que, si en un futuro quiere cambiar cualquier elemento de su estilo CSS, debe ir uno por uno cambiando el estilo CSS en todos los documentos HTML donde lo haya utilizado. Para poder poner solución a este problema, siempre se aconseja utilizar el código CSS de manera externa a su documento HTML. De hecho, esta es la manera profesional de hacerlo. Lo único que debe hacer es crear un archivo CSS externo y para utilizarlo en su documento HTML, tan solo debe referenciar el documento CSS median un link. Gracias a esto si en un futuro quiere, por ejemplo, cambiar el color del texto del estilo que programa en CSS, tan solo debe ir a su archivo CSS y cambiarlo, y automáticamente se empleará el nuevo estilo en su documento HTML. Vea un sencillo ejemplo:

Documento HTML

```
<!DOCTYPE html>

<html>

<head>

<style type="text/css">

@import url("url de la hoja de estilo .css");

</style>

<meta charset="utf-8">

<title>Ejemplo de código HTML y CSS</title>
```

```
</head>

<body class="nombreEstilo">

<p>Estamos importando una hoja de estilo .CSS</p>

</body>

</html>
```

Documento CSS

```
.nombreEstilo{

Font: bold 14px Verdana, Arial;

Color:"green";

}
```

Cuando utiliza código CSS, puede emplear diferentes técnicas de diseño web. Vea a continuación algunas de ellas:

**Centrar página con ancho fijo.** Hoy día se sabe que los diferentes dispositivos poseen pantallas de tamaños totalmente diferentes, es por ello que, si quiere diseñar una web que sea totalmente accesible para usuarios de ordenador, de móvil o de Tablet, debe tener en cuenta utilizar esta técnica de ancho fijo. Así, controla el ancho de la propia página, aunque las pantallas donde se muestre sean diferentes.

Antiguamente, esta técnica se realizaba con un div de la siguiente manera:

```
<div align="center">Contenido</div> y se añadía a una tabla.
```

Sin embargo, actualmente CSS aporta tres maneras diferentes de realizar este trabajo. El primer método es centrar un elemento, que normalmente estará contenido en un div, con un ancho fijo y, posteriormente, ajustar los márgenes izquierdo y derecho en modo auto. Vea el ejemplo:

```
Div#page{

Width: 400px;

Margin-left: auto;

Margin-right: auto;}
```

Lo bueno de este método, es que funciona para todos los navegadores que se guían por los estándares establecidos del CSS. Sin embargo, los navegadores antiguos no soportarán este método.

El segundo método que verá es menos profesional, pero es efectivo para centrar toda la página. Para ello, utiliza la propiedad text-align dentro de la etiqueta body. También, otra opción es utilizar esta técnica tomando la propiedad de texto, utilizándola para centrar cualquier elemento.

Sin embargo, este método tiene un problema y es que el alineamiento horizontal es heredado, es decir, todo el texto que alberga su página está centrado en cajas de elemento. Por ello, debe especificar en el código que se ignore el centrado heredado y, para ello, se indicará en la propiedad align que el texto se alinee a la izquierda, ya que lo único que quiere que se centre es el propio recipiente que contiene el texto, no el texto en sí.

En el código que va a ver de ejemplo, va a ver que la etiqueta body tiene un selector universal que se indica con el asterisco. Este selector universal selecciona todos los elementos que aparecen en el cuerpo de su documento HTML y ajustará la alineación del texto a la izquierda. Por otro lado, margin-left y margin-right están abreviados por margin. Así, se reduce la cantidad de código manteniendo el documento más limpio y sencillo de modificar. Vea el ejemplo:

```
Body{text-align: center;}
```

```
Body*{text-align:left;}
```

```
Div#page{
```

```
Width: 400px;
```

```
Margin: 0 auto;}
```

El tercer y último método consiste en utilizar márgenes negativos para centrar bloques contenedores de manera eficaz en todos los navegadores que soporten el posicionamiento absoluto. Primeramente, posiciona la página de manera absoluta, por lo que, su lado izquierdo queda posicionado al 50% del contenedor inicial. Después, aplica un valor negativo al margen izquierdo que es el encargado de mover la página a la izquierda a la mitad de su anchura. Así, es como se termina alineando el centro del bloque con el centro de la ventana. Vea el ejemplo:

```
Div#page{
```

```
Position: absolute;
```

```
Left: 50%;
```

```
Width: 550px;
```

```
Margin-left: -250px;
```

}

**Diseño multicolumna.** Los diseños multicolumna parten de las tablas en HTML. Actualmente, puede crear diseños multicolumna a partir de código CSS. Para ello, utiliza las opciones de posicionamiento absoluto. Con CSS puede hacer todo tipo de diseños de columnas utilizando todos los componentes que ofrece CSS con los que puede modificar medidas, colores, posiciones, etc.

Puede hacer diseños de dos y tres columnas, aunque en los diseños de tres columnas debe planificar mejor el diseño, ya que, es más complejo que el de dos columnas.

**Bloque con esquinas redondeadas.** Actualmente este tipo de diseño es algo esencial. Esto sucede porque en el diseño web todo es básicamente rectangular y no hay posibilidad de insertar elementos con esquinas redondeadas por defecto. Antes de que existiera el CSS la única opción de conseguir este efecto era utilizar tablas con nueve celdas. Sin embargo, actualmente puede conseguir este efecto utilizando CSS sin necesidad de crear ninguna tabla. Para crear este tipo de efectos hay múltiples opciones, pero va a ver la opción más sencilla. Este método debe tomar cuatro archivos de imagen, es decir, uno por esquina, aplicándolo como imágenes de fondo a cuatro elementos de marcado. Por lo tanto, los archivos que tendría serían los siguientes:

esquinaSuperiorDerecha.jpg

esquinaSuperiorIzquierda.jpg

esquinaInferiorDerecha.jpg

esquinaInferiorIzquierda.jpg

Esta técnica consiste en que el marcado tenga aseguradas cuatro elementos disponibles para reemplazar la imagen de fondo. Esto se suele denominar ganchos en el marcado donde en esos ganchos puede aplicar los estilos. En el caso en el que la estructura del documento tenga menos de cuatro elementos, será necesario añadir algún div y, así, conseguir el número necesario de ganchos en su documento. Vea el siguiente ejemplo:

```
<div class="caja">
 <div class="top">
 <div></div>
 </div>

 <div class="contenido">
 <h1>Título inicial</h1>

 <p>Texto</p>
```

---

```
</div>

<div class="bottom">

<div></div>

</div>

</div>
```



TOME NOTA

**Añadir elementos vacíos en un documento daña la integridad semántica del mismo, por lo que esto suele ser una mala praxis. Sin embargo, para insertar algunos efectos visuales, debe utilizar elementos vacíos.**

**Reemplazo de texto por imágenes de fondo.** Si intenta sustituir un texto real, por ejemplo, un elemento h2, por una imagen, lo que ocurrirá es que el texto se eliminará completamente. Podría ponerle a la imagen un texto alternativo, pero seguiría sin ser una solución, ya que, al hacer esto el documento original se habría modificado. Además, como ha comentado en apartados anteriores, lo más profesional es utilizar CSS para añadir efectos decorativos al documento HTML y, de esta manera, lo deja intacto, lo que aportará mayor facilidad de modificación del mismo en un futuro.

Por otra parte, el código CSS sufrió en 2006 una modificación en las técnicas de reemplazo de imágenes que permitían sustituir elementos de texto. Sin embargo, el elemento de texto sigue estando en el documento original, aunque en pantalla solo vea la imagen de sustitución. Aunque existan esas técnicas mencionadas anteriormente, actualmente no existe una solución ideal en el código CSS para el reemplazo de imágenes, ya que, estas técnicas tan solo son soluciones temporales. Esto sucede porque las técnicas que hoy por hoy se pueden implementar, son técnicas que dan por hecho que el usuario es capaz de leer el contenido en las imágenes, aunque éstas tengan el texto oculto. Esto quiere decir que, si el usuario tiene una conexión lenta de Internet o tienen activado el CSS con las imágenes desactivadas, estarán coartados por la propia solución. En cambio, con CSS 3 sí que es posible utilizar la técnica de reemplazo por imágenes. Para realizarlo se realizará de la siguiente manera:

```
H2{content:url(imagen.jpg);}
```

Sin embargo, el problema está en que los navegadores, hoy por hoy, no soportan esta funcionalidad, por lo tanto, aún no es viable.

**Rollovers en CSS.** El rollover es un efecto visual que consiste en que cuando el puntero del ratón se mueve por encima de determinado elemento, este cambia. Los rollovers son muy importantes, ya que, se usan para indicar que un vínculo o botón es interactivo y que el usuario puede utilizarlo.

Antiguamente, para crear estos efectos era necesario utilizar JavaScript, sin embargo, hoy por hoy puede realizarlo cómodamente con CSS. Para ello, tan solo debe utilizar el selector de pseudoclase :hover.



TOME NOTA

**Se debe saber que el selector :hover solo tiene soporte en vínculos en el navegador Internet Explorer 6 o anteriores. Sin embargo, a partir de Internet Explorer 7 sí se amplía el soporte.**



TOME NOTA

**Para que funcione este elemento, deberán aparecer en un orden específico los selectores de pseudoclase. Siendo el orden el siguiente: :link, :visited, :hover, :active.**

A continuación, va a ver un ejemplo de rollover de texto:

```
/*Enlace no visitado*/

a :link{

color:blue;

}

/*Enlace visitado*/

a :visited{

color: purple;

}

/*Cursor encima del enlace*/
```

```
a :hover{
color: green;
}

/*Enlace seleccionado*/

a :active{
color: red;
}
```

**Rollover de imagen.** Este tipo de rollover funciona de igual manera que el anterior, sin embargo, la diferencia es que cuando el puntero del ratón se sitúa sobre una imagen lo que cambiará es el elemento background-image. A continuación, va a ver un ejemplo en el que una hoja de estilo establece una imagen de fondo a todos los hipervínculos del documento. En este ejemplo, va a ajustar el elemento a para que aparezca en pantalla como un rectángulo en el que las dimensiones coincidirán con el tamaño de la imagen. Por otra parte, el elemento a:hover aplicará una imagen diferente para que así, cuando el usuario sitúe el puntero del ratón encima de la imagen, esta cambie. Vea el ejemplo:

```
boton{

Display: block;

Width_ 183px;

Height: 40px;

Text-indent: -9999px;
}

.boton a{

Display:block;

Width:184px;

Height:41px;

Background: transparent url(http://rutaDeLalmagen/boton.png) no-repeat top left;

Outline: none;
```

}

.boton a:hover{

Background-position:0 -40px;

}

**Rollover sin carga previa.** Este tipo de rollover está basado en que todos los estados del rollover se ubican en una misma imagen, lo único que cambia para cada estado del enlace es el elemento background-position. Es así como evita las recargas y las cargas previas de las imágenes. Esto hace que su página sea más rápida y, por lo tanto, aporte una mejor experiencia de usuario.



TOME NOTA

*Al aplicar imágenes de fondo y rollovers puede ocurrir que se presente un parpadeo en el navegador de Internet Explorer. La solución que puede aplicar es introducir una imagen de fondo al vínculo a y al elemento que lo contiene.*

**Navegación basada en listas.** La presencia de barras de navegación horizontales siempre ha sido muy frecuente en el diseño web. Este tipo de barras de navegación, antiguamente, se creaban con múltiples enlaces que presentaban textos adyacentes. Sin embargo, esta técnica no aportaba mucho sentido al marcado del documento. Es por ello, que actualmente las opciones de navegación se marcan como lista en el documento. Para ello, utiliza CSS y aplicará el marcado semántico correcto al elemento. Además, tiene dos maneras de convertir una lista en una barra de navegación horizontal. La primera manera es hacer que los elementos de la lista aparezcan de manera horizontal, es decir, en línea en vez de uno debajo del otro. La segunda manera es utilizar la opción de flotamiento para posicionar los elementos de la lista de manera horizontal.

### 2.3. Trabajar con marcos

Cuando comienza a diseñar una página web, lo primero que debe hacer es planear cómo van a distribuirse los elementos. Por ejemplo, debe plantear qué elementos va a utilizar en su web, si menús verticales u horizontales. Si va a utilizar barra buscadora, banners, sliders, si quiere repartir las secciones con imágenes. Qué tipos de vínculos va a utilizar y adónde van a llevar. Los tipos de botones que utiliza y qué estilo tendrán. Obviamente, todo esto derivará del tipo de web que vaya a crear. Por ejemplo, no es lo mismo una web para un dentista, que una web para vender productos alimenticios.

Para realizar la distribución de las secciones en la web, puede utilizar, por ejemplo, tablas donde a cada celda puede aplicarle diferentes dimensiones según vaya interesando. Aunque, este método sea muy flexible a la hora de organizar la página, realmente esta no es la mejor forma para dividir la página por secciones, ya que, tiene bastantes inconvenientes.

Como alternativa al uso de las listas, puede aplicar marcos a su página. Con los marcos puede realizar diferentes secciones en las cuales puede dividir la ventana incluyendo un documento diferente en cada una de las secciones. Este tipo de elementos solo es posible de utilizar en los navegadores de tipo visual, ya que, son los únicos que cuentan con una ventana que se puede dividir en diferentes secciones. El hecho de utilizar marcos requiere más tiempo de programación, ya que, debe crear varios documentos por separado para, posteriormente, insertarlos en cada una de las secciones y así componer su página web. Sin embargo, el uso de marcos también tiene inconvenientes como, por ejemplo, la dificultad de impresión de documentos, en el caso que quisiera imprimir parte de la web o, por ejemplo, que no es compatible con los navegadores no visuales. Por otra parte, las cosas buenas que aporta el uso de marcos es que permiten diseñar sitios web de manera sencilla y flexible.



*El uso de marcos tiene más inconvenientes que beneficios. De hecho, hay comunidades en Internet que están en contra del uso de los marcos.*

TOME NOTA

Por otra parte, HTML aporta los frames que son elementos que permiten, también, partir la página en distintas secciones independiente, permitiendo colocar diferentes páginas dentro de la página principal. Los frames son elementos compatibles con Internet Explorer y Netscape a partir de su versión 2.0 en adelante.

El origen de los frames se remonta al navegador Netscape, el cual ya no existe, donde los frames eran elementos propios de este navegador. Al ver lo potente que era este recurso, se extendió su uso a otros navegadores. Por ello, Internet Explorer, entre otros, tardó muy poco tiempo en incluirlos también como elementos compatibles y así, diversos navegadores pudieron competir con Netscape. Por lo tanto, en la versión HTML 4.0 fue cuando se incluyó los frames como elemento propio y, por lo tanto, como etiqueta HTML.

Como ha visto anteriormente, los frames permiten crear secciones en las que insertar diferentes documentos en la misma página. Cada sección, es decir, cada frame es independiente uno del otro y es por ello que cada archivo que se muestra, ha tenido que ser codificado previamente a la inserción del mismo en el frame. Además, cada frame puede contener propiedades específicas que serán especificadas en el código HTML. Por último, debido a que cada frame es independiente uno del otro, también, contendrá barras de desplazamiento horizontal y vertical independientes de cada frame.

Seguramente, haya topado con cientos de páginas web que utilizan frames. Estos elementos, suelen aparecer en una parte de la ventana, y en ellos se pone la barra de navegación que se suele encontrar fija. Es así, como el usuario puede acceder al menú desde cualquier sitio de una web. La principal ventaja del uso de frames es que al ser estos independientes entre sí, permite navegar por

una página web desde una barra de navegación fija y siempre visible desde cualquier parte de la web, y todo ello sin tener que recargar la web en cada uno de los puntos visitados de la misma.

Frame en inglés significa marco o cuadro y, como ha visto anteriormente, es un elemento implementado por el navegador Netscape. Este elemento permite dividir la pantalla en las áreas que desee y cada una de esas áreas es diferente entre sí. Estas áreas funcionan como si de ventanas independientes se tratase. Nunca debe confundir los frames de las tablas, aunque, a veces, puedan presentarse de manera muy similar. Debe saber que la celda de una tabla es un área totalmente estática mientras que un frame es un área totalmente dinámica y en la que puede emplear propiedades de HTML para modificarla a su antojo y diferenciarlas entre sí.

Cuando crea una página con el elemento frame, no podrá aparecer el elemento body, ni ningún elemento que ha visto hasta ahora antes de la etiqueta framset, que es la que indica la creación de un frame. Si esto no sucede así, el frame y todo lo que se contenga en su interior se ignorará a la hora de la ejecución del código.

Como ha mencionado anteriormente, los frames aportan mucha flexibilidad y dinamismo a la página web. De hecho, si quiere realizar páginas web de una complejidad superior, los frames se volverán elementos insustituibles. Pues, son los únicos elementos capaces de ser moldeados tan específicamente. Es por ello, que puede parametrizarlos conforme a su tamaño, cantidad de áreas, si el usuario puede redimensionar las áreas o éstas son fijas, si contienen barras de scrolling o no, puede también anidar frames unos en otros y relacionar sus contenidos.

En cambio, los frames son totalmente incompatibles con los navegadores no visuales. Esto quiere decir, que no pueden ser utilizados para usuarios invidentes, pues los frames, como ha visto anteriormente, solo son compatibles con navegadores visuales. Por lo tanto, los navegadores no visuales bloquean el contenido de los frames. Una solución para este hecho, es crear dos versiones de la misma web. Una versión con frames y otra sin frames, y así puede crear una web que permita su visualización a todos los tipos de usuarios.

A continuación, va a ver una tabla donde puede aprender las diferentes características de los frames:

Frames	Etiquetas	Descripción
Altura de los márgenes	<frame marginheight=>	Margen superior e inferior
Anchura de los márgenes	<frame marginwidth=>	Margen izquierdo y derecho
Denominación de frame	<frame name="X"   _blank   _self   _parent   _top>	Se utiliza para saber cómo se abre el frame
Documento a mostrar	<frame src="url">	Documento que se muestra dentro de un frame
Color del borde	<frameset bordercolor="#fffff"	Color en hexadecimal o por nombre
Borde del frame	<frameset frameborder= "yes no"	Inserta un borde al frame
Anchura de borde	<frameset border=>	Anchura expresada en píxeles
Anchura de columnas	<frameset cols="*>	Tamaño relativo
Anchura de columnas	<frameset cols=",,>	Tamaño en píxeles o

		porcentaje
Altura en filas	<frameset rows="*>	Altura relativa
Altura en filas	<frameset rows=",,,>	Píxeles o porcentaje
Barra de desplazamiento	<frame scrolling="yes no auto">	Insertar barra de desplazamiento
No redimensionable	<frame noresize>	Frame con tamaño fijo
Sin marco	<noframes>	Se utiliza para navegadores no compatibles con marcos

Cuando crea una página web con frames, debe aparte crear varios archivos HTML que incrustará en las diferentes áreas de los frames que, a su vez, se encontrarán en una página principal. Por lo tanto, primero debe crear el archivo principal y después cada uno de los archivos HTML.

Por ejemplo, si quiere crear una página web dividida en dos marcos, un marco superior y un marco inferior, donde cargar el archivo superior.htm en el marco superior y el archivo inferior.htm en el marco inferior. Para hacer posible la ejecución de ambos archivos, debe crear un elemento que será el encargado de invocar ambos archivos. Este elemento le asignará a cada archivo .htm el lugar donde deben aparecer en la página principal. Vea a continuación el código para crear este elemento:

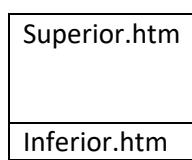
```
<frameset rows="80,*">

<frame name="superior" src="superior.htm">

<frame name="inferior" src="inferior.htm">

</frameset>
```

Utilizando este código los archivos quedarán de la siguiente manera:



Además, como puede observar en el código se utilizan las etiquetas <framset> que actúan de la misma manera que las etiquetas <html> indicando que ese trozo de código pertenece a la creación de los frames.

Para indicar el tamaño de los marcos, se utiliza la propiedad rows que como puede observar en el código tiene un valor de 80 que indica el alto del frame, que tiene un valor de 80 píxeles de alto. Por otro lado, tiene el valor \* que significa que el resto del espacio debe asignarse al marco inferior. Si hubiera querido indicar la altura en porcentajes, debería haber utilizado la siguiente instrucción:

```
<frameset rows="30%,*>
```

Una vez haya establecido las etiquetas <frameset> y </frameset>, dentro de ambos será donde establezca los nombres y los documentos que deben ser invocados dentro de los marcos que haya creado previamente. Para ello, debe haber dado un nombre a cada marco, por ejemplo, name="superior", e indica el archivo que quiere insertar en él estableciéndolo con la instrucción src="superior.htm". Es por ello, que los archivos deberán haber sido creados antes de crear los diferentes marcos.

Para finalizar con este ejemplo, debe tener muy en cuenta el orden con el que el navegador interpreta las diferentes instrucciones, ya que, si invierte el orden de los frames, el navegador los posicionará al revés de como realmente quiere que se muestren. Vea el siguiente código:

```
<frameset rows="80,*">

<frame name="inferior" src="inferior.htm">

<frame name="superior" src="superior.htm">

</frameset>
```

En este ejemplo, los documentos se posicionaría de manera totalmente inversa estando el documento inferior.htm en el marco superior y el documento superior.htm en el marco inferior.

A continuación, va a ver un ejemplo en el que crea dos marcos alineados verticalmente. Para ello, va a modificar la instrucción rows por cols. Vea el siguiente ejemplo:

```
<frameset cols="100,*">

<frame name="izquierda" src="izquierda.htm">

<frame name="derecha" src="derecha.htm">

</frameset>
```

Este código mostraría dos marcos de la siguiente manera:

Izquierda.htm	Derecha.htm
---------------	-------------

Como puede observar, para crear dos marcos alineados verticalmente, ha utilizado la instrucción cols, en vez de rows.

Como vio anteriormente, los navegadores antiguos no soportaban los frames, por lo tanto, para solucionar este problema existe una instrucción con las etiquetas <noframes> que le indica a este tipo de navegadores la imposibilidad de visualizar la página correctamente debido a la incompatibilidad de estos con los marcos. Para realizar esta acción debe utilizar el código que se muestra a continuación:

---

<noframe>

<html>

<body>

Advertencia. Su navegador no soporta el uso de frames. Si desea visualizar el contenido de esta web, necesitará utilizar un navegador que lo soporte.

</body>

</html>

</noframe>

Volviendo al uso de los frames, es posible crear divisiones en los frames de columnas y filas simultáneamente. Esto ayudará a crear una estructura de áreas más complejas que las que ha visto anteriormente. A continuación, va a visualizar diferentes tipos de ejemplos con su código y con la visualización gráfica de cómo se posicionarían los diferentes marcos. Vea los ejemplos:

Vea el código para crear la siguiente división:

Superior	
Izquierda	Derecha

```
<frameset rows="100,*">
 <frame name="superior" src="superior.htm">
 <frameset cols="150,*">
 <frame name="izquierda" src="izquierda.htm">
 <frame name="derecha" src="derecha.htm">
 </frameset>
</frameset>
```

Ahora, vea el código para la siguiente división:

Izquierda	Superior
	inferior

```
<frameset cols="120,*">

<frame name="izquierda" src="izquierda.htm">

<frameset rows="100,*">

<frame name="superior" src="superior.htm">

<frame name="inferior" src="inferior.htm">

</frameset>

</frameset>
```

A continuación, vea la siguiente disposición:

Izquierda	Superior
	Centro
	inferior

```
<frameset cols="120,*">

<frame name="izquierda" src="izquierda.htm">

<frameset rows="20%, 60%, 20%, *">

<frame name="superior" src="superior.htm">

<frame name="centro" src="centro.htm">

<frame name="inferior" src="inferior.htm">

</framset>

</frameset>
```

Vea el siguiente ejemplo:

```
Superior Derecha

| inferior |

<frameset cols="75%, 25%">

<framset rows="20%, 80%, *">
```

```
<frame name="superior" src="superior.htm">

<frame name="inferior" src="inferior.htm">

</frameset>

<frame name="derecha" src="derecha.htm">

</framset>
```

Observe la siguiente posición:

superior	derechaTotal
izquierda	derecha

```
<frameset cols="75%, 25%">

<framset rows="20%, 80%*>

<frame name="superior" src="superior.htm">

<framset cols="20%, 80%*>

<frame name="izquierda" src="izquierda.htm">

<frame name="derecha" src="derecha.htm">

</frameset>

</framset>

<frame name="derechaTotal" src="derechaTotal.htm">

</framset>
```

Observe la siguiente disposición:

superiorIzquierda	superiorDerecha	
Izquierda	Centro	derecha

```
<frameset cols="75%, 25%">

<framset rows="20%, 80%*>
```

```
<frame name="superiorIzquierda" src="superiorIzquierda.htm">

<frame name="centro" src="centro.htm">

</frameset>

<framset rows="24%, 76%">

<frame name="superiorDerecha" src="superiorDerecha.htm">

<frame name="derecha" src="derecha.htm">

</frameset>

</frameset>
```

Pase a ver el siguiente ejemplo:

IzquierdaSuperior	centro
IzquierdalInferior	

```
<frameset cols="25%, 75%">

<framset rows="80%, 20%">

<frame name="izquierdaSuperior" src="izquierdaSuperior.htm">

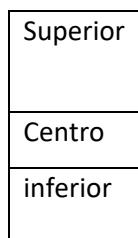
<frame name="izquierdalInferior" src="izquierdalInferior.htm">

</frameset>

<frame name="centro" src="centro.htm">

</frameset>
```

Vea la siguiente disposición:



```
<frameset rows="20%, 60%, 20%">

<frame name="superior" src="superior.htm">

<frame name="centro" src="centro.htm">

<frame name="inferior" src="inferior.htm">

</frameset>
```

Y, por último, va a ver el siguiente ejemplo:

Izquierda	Centro	derecha
-----------	--------	---------

```
<frameset cols="20%, 60%, 20%">

<frame name="izquierda" src="izquierda.htm">

<frame name="centro" src="centro.htm">

<frame name="derecha" src="derecha.htm">

</frameset>
```

En el caso en que quiera eliminar el borde gris que aparece en los marcos, debe utilizar el siguiente código:

```
<frameset cols="20%, 60%, 20%" border=0>
```

Por otra parte, si quiere que los marcos tengan un tamaño fijo y que el usuario no pueda redimensionarlos, utiliza el código siguiente:

```
<frame name="superior" src="superior.htm" noresize>
```

Si quiere eliminar las barras de desplazamiento de los marcos, utiliza el código que se muestra a continuación:

```
<frame name="superior" src="superior.htm" scrolling="no">
```

En cambio, si quiere que siempre se muestren las barras de desplazamiento, utiliza el siguiente código:

```
<frame name="superior" src="superior.htm" scrolling="yes">
```

Por último, si solo quiere mostrarlas cuando sea necesario, utiliza este código:

```
<frame name="superior" src="superior.htm" scrolling="auto">
```

También, puede regular la distancia que hay desde el contenido del marco hacia el margen superior del mismo y a los márgenes izquierdo y derecho con el siguiente código:

```
<frame name="superior" src="superior.htm" marginheight=3 marginwidth=6>
```

Puede cargar una página en un marco diferente al que se encuentra el enlace real de esta página. Para ello, debe hacer referencia al nombre que haya asignado a determinado marco. Cuando se refiere al nombre de los marcos, no se refiere al nombre del archivo, es decir, al src sino al name. Vea un ejemplo:

```
<frame name="superior" src="superior.htm"> El nombre asignado a este marco es superior, no superior.htm.
```

Para realizar este ejemplo, va a tomar una página dividida en dos marcos alineados horizontalmente, es decir:

Izquierda	derecha
-----------	---------

```
<frameset cols="20%,60%,20%">
```

```
<frame name="izquierda" src="izquierda.htm">
```

```
<frame name="derecha" src="derecha.htm">
```

```
</frameset>
```

Imagine que en el marco izquierdo tiene un enlace y quiere cargar la página a la que lleva ese enlace en el marco derecho. Bien, pues si el código que inserta en el marco izquierdo fuera el siguiente:

```
Pulse aquí
```

Este código indica al navegador que cuando el usuario pinche el enlace, cargue la página en el mismo marco izquierdo en el que está insertado el propio enlace, ya que, el navegador carece de las instrucciones necesarias que le indiquen dónde debería cargar la página. Sin embargo, si quiere indicarle al navegador que cuando el usuario pulse el enlace que se ubica en el marco izquierdo, abra la página correspondiente en el marco derecho, debe aplicar el siguiente código donde con la instrucción target, indica al navegador dónde debe abrir la página. Véalo:

```
Pulse aquí
```

Además, también puede utilizar la propiedad target para abrir un enlace en otra página. Por lo tanto, al abrir el enlace en otra nueva página, este ocupará toda la pantalla completa en donde solo y únicamente se visualizará el contenido del enlace. Vea cómo hacerlo en el siguiente ejemplo:

<a href="nuevaPagina.htm" target="\_parent">Pulse aquí</a>

Por otra parte, si utiliza el siguiente código:

```
<base target="_top">
```

En el head del documento HTML, todos los enlaces que tenga en la misma página van a eliminar los marcos presentes, sin tener que clicar los enlaces de uno en uno. En cambio, si quiere ejecutar más de un enlace al mismo tiempo, la página debe de estar dividida mínimo en tres frames.

En el siguiente ejemplo, va a tomar una página dividida en tres áreas de la siguiente manera:

Izquierda	Superior
	Inferior

Esta división la creará con el siguiente código.

```
<frameset cols="120,*">

<frame name="izquierda" src="izquierda.htm">

<frameset rows="100,*">

<frame name="superior" src="superior.htm">

<frame name="inferior" src="inferior.htm">

</frameset>

</frameset>
```

Una vez tiene creada la estructura, su finalidad es insertar en el área izquierda un enlace que al clicar abra dos páginas diferentes de manera simultánea, una en el área superior y otra en el área inferior. Para ello, debe introducir código JavaScript. Primeramente, debe insertar este código que vea a continuación, entre las etiquetas head de su documento HTML:

```
<head>

<script language="JavaScript">

Function cargarDos(page2,page3){

Parent.superior.location.href=page2;

Parent.inferior.location.href=page3;
```

```
}
```

```
</script>
```

```
</head>
```

El segundo paso será introducir el siguiente código entre las etiquetas body. Este código indicará dónde se insertará el enlace:

```
<body>
```

```
<form name="botones">
```

```
<input type="button" value="Pulse aquí"
```

```
Onclick="cargarDos('pagina1.htm','pagina2.htm')">
```

```
</form>
```

```
</body>
```

Al insertar estos elementos no debe olvidarse de configurar los bordes y las barras de desplazamiento del mismo. Si, por ejemplo, desea que las dimensiones de ancho y largo de los marcos tengan un tamaño fijo, debería eliminar los bordes que los delimitan. Para realizar esta acción tan solo tiene que añadir la propiedad frameborder="0" a los tres frames que aparecen en el documento HTML. Así, consigue darle homogeneidad al aspecto de la página principal.

Aun habiendo eliminado los bordes de separación de los frames, si desplaza el puntero del ratón hacia donde se encontraría el borde que ha eliminado, el puntero del ratón cambiará de forma y será capaz de cambiar incluso las dimensiones del propio frame. Esto sucede porque al eliminar los bordes, tan solo ha eliminado el color del borde, pero no ha indicado que quiere que el borde sea de tamaño fijo. Para ello, tan solo debe añadir el atributo noresize a los elementos frame que no quiera que sean redimensionables. Si pone este atributo a los marcos, que es un atributo booleano y no necesita que le indique ningún tipo de valor, al pasar ahora el puntero del ratón por los bordes no cambiará de forma.

Además, puede también controlar si se muestran o no las barras de desplazamiento en cada marco. Si, por ejemplo, un usuario utiliza las opciones del navegador que permite agrandar el tamaño de la letra de la página, estas barras de desplazamiento aparecerán. Si quiere impedir que aparezcan las barras de desplazamiento, tan solo, debe incluir la propiedad scrolling="no" a los frames correspondientes.

Con todos estos recursos, habrá creado una página principal totalmente homogénea donde visualmente no existe ninguna separación de contenido, es decir, que no parece estar parcheada. Sin

embargo, sabrá que en la página hay marcos porque puede desplazarse por el cuerpo de la misma mientras las áreas del pie y la cabecera se mantienen estáticas.



TOME NOTA

***Debe diferenciar entre las dimensiones de los marcos contenedores con las dimensiones de los elementos que se contienen en ellos. Es decir, los elementos que se ubican dentro de los marcos pueden tener un tamaño fijo, mientras que el contenedor puede tener un tamaño variable***

### 2.3.1. Dónde utilizar los marcos

Si en su página web tiene enlaces que apuntan a sitios externos de su propia página, lo que sucede es que estos enlaces externos aparecen en el interior de su propio marco, y esta práctica no es correcta. También debe recalcar que estas url que apuntan a páginas externas, no aparecen en la barra de direcciones, sino que en esta sigue apareciendo la url de su propia página.

En estos casos lo más correcto es utilizar el atributo target para indicar al navegador dónde debe abrir esa página externa. El atributo target tiene diversas opciones para indicar cómo y dónde se debe abrir un enlace determinado. Por ejemplo, la opción target="\_top", abrirá la página externa la cual sustituirá todos los marcos existentes en su web y, por lo tanto, aparecerá la página externa como documento principal en el navegador.

Por otro lado, con target="\_blank" su página web quedará abierta en una pestaña del navegador, por lo que, quedará totalmente intacta, y aparte el documento externo se abrirá en una nueva ventana. Si utiliza la opción target="\_self", lo que ocurrirá es que la página externa cargara en la misma ventana que está contenida su página web. Para finalizar, si utiliza target="\_parent", la página externa cargará donde se encuentra el contenido actual.



TOME NOTA

***Siempre que diseña una página web con marcos, no debe olvidar de añadir el atributo target a cada hiperenlace que aparezca en su web para indicar al navegador dónde y cómo debe abrir el documento enlazado.***

Además, también tiene la opción de insertar en su documento HTML un documento externo utilizando un marco embebido. Los marcos embebidos son los conocidos en HTML como iframes. Este tipo de marcos aportan la funcionalidad de visualizar un documento externo dentro del documento principal sin dividir la página en diferentes zonas. De hecho, ha visto infinidad de veces el uso de iframes aunque no supiera realmente de que se trataba de estos elementos. Por ejemplo, las típicas páginas de Internet donde puede visualizar un vídeo de Youtube utilizan iframes para ello.

Los iframes pueden aparecer en cualquier área de su documento HTML. Para utilizar este atributo debe utilizar las etiquetas tanto de apertura como de cierre obligatoriamente. El contenido que puede albergar dentro de los iframes puede ser desde texto hasta imágenes o vídeos.

A continuación, va a ver algunos atributos de la etiqueta iframe:

- **Id.** Este atributo asigna un nombre identificador al iframe. El id que asigne a un iframe debe de ser único, ya que, el id servirá para indicar el propio elemento. Si tiene dos iframes con el mismo id será imposible diferenciarlos y, por lo tanto, no funcionarán correctamente.
- **Class.** Este atributo asigna un nombre de clase al elemento iframe. Los nombres de clase, como debe recordar, se utilizan sobre todo en las hojas de estilo CSS para indicar a qué elemento o elementos está asociado dicho estilo.
- **Style.** Este atributo se utiliza para introducir valores en código CSS, es decir, aplicar estilos directamente dentro del elemento. Aunque, como ya ha visto anteriormente, siempre es mejor poner los estilos CSS en archivos externos. Esto conllevará una mejor lectura del documento y una más sencilla modificación en un futuro.
- **Title.** Este atributo, al igual que pasa en el head del HTML, se utiliza para escribir un título en el documento. Para ello, insertará un pequeño texto que describa de forma precisa la información que se está presentando en el iframe. Los títulos de los iframes se mostrarán al dejar el puntero del ratón unos instantes en el propio recuadro.
- **Src.** Este atributo permite apuntar a un documento externo tal y como ha hecho anteriormente con los frames.
- **Scrolling.** El atributo scrolling tiene diversas opciones. Por lo general, es el que permite mostrar o no mostrar barras de desplazamiento en el recuadro. Tiene la opción auto que solo muestra barras de desplazamiento cuando sea necesario. La opción yes siempre mostrará las barras de desplazamiento y la opción no, nunca mostrará la barra de desplazamiento.
- **Marginwidth.** Este atributo asigna la distancia entre el contenido del iframe y los márgenes izquierdo y derecho de este.
- **Marginheight.** Por otra parte, este atributo asigna la distancia entre el contenido del iframe y los márgenes superior e inferior del iframe.
- **Width.** Este atributo asigna la distancia entre el margen derecho e izquierdo del iframe.
- **Height.** Este atributo asigna la distancia entre el margen superior e inferior del iframe.
- **Name.** Este atributo es el que permite identificar el marco, mediante este atributo puede utilizar enlaces y formularios como valor del atributo, que ha visto anteriormente, target.
- **Align.** Este atributo está obsoleto y, por lo tanto, no se recomienda su uso. Sin embargo, permite la alineación horizontal del contenido del marco. Las opciones de uso de este atributo son left, que permite alinear el contenido a la izquierda, right que permite alinear el contenido a la derecha, center que permite alinear el contenido al centro y justify que permite alinear el contenido a ambos lados.

Como ha podido observar muchos de los atributos que se han mostrado son similares a los atributos de los frames como, por ejemplo, framborder, marginheight, etc. Y funcionan también de la misma manera que en los frames. Sin embargo, el atributo más importante es src, ya que, es el que posee la función de apuntar a un sitio externo. En el atributo src indicaría el sitio determinado que quiere que se muestre dentro de su iframe y, así, mostraría una ventana embebida en su documento inicial.

---

Por otro lado, los atributos width y height permiten asignar el ancho y el alto del iframe, en cambio el atributo align permite alinear el iframe a un margen y dejarlo como elemento flotante. Debe tener en cuenta que todos estos atributos puede modificarlos desde el código CSS.



***Cabe destacar que el documento al que apunta el atributo src no debe ser obligatoriamente un documento HTML, puede ser perfectamente un archivo de texto, una imagen, un vídeo, etc.***

TOME NOTA

El uso de marcos permite organizar la información que alberga una página web facilitando la lectura y la interacción con la misma. Al dividir una web en diferentes áreas, denomina a estas áreas como marcos o frames. A su vez, a donde están contenidos estos marcos se le denomina página de marcos.

Normalmente, el uso de marcos ayuda y facilita la lectura y compresión de la información de la web porque, normalmente, una web se compone de varias páginas. Por lo que, gracias a los marcos puede visualizar los diferentes contenidos sin que se abran múltiples ventanas y ya no sepa dónde queda la página principal. Además, los marcos también se pueden utilizar para crear una tabla de contenidos que queda estable, aunque se desplace por las múltiples páginas que contiene la web. En el mundo del desarrollo web puede encontrar diversos tipos de diseños web con marcos, ya que, estos se utilizan para mostrar cualquier tipo de contenido como, por ejemplo, mostrar las fotos de un álbum de fotos digital o mostrar una tabla con los libros de una biblioteca digital.

Las páginas web con marcos se crean mediante código HTML y, al igual que puede moldear los marcos cuando crea un documento Word con marcos, puede hacerlo en su página web. Es decir, los frames y iframes son totalmente flexibles. Obviamente, cuantos más marcos añada, más compleja será la estructura. Posteriormente a la creación de los marcos, es cuando se pone a los mismo los vínculos e indica a dónde se debe dirigir en el caso en que el usuario interaccione con los propios vínculos.

Los marcos que crea por defecto mostrarán un borde gris y, además, contendrán barras de desplazamiento. Pero como ya vio anteriormente, puede modificar estos atributos tanto como quiera. Puede eliminar los bordes e incluso eliminar las barras de desplazamiento y utilizar el atributo noresize para que el usuario sea incapaz de cambiar el tamaño de los marcos. Debe recordar que no debe abusar tampoco del uso de los marcos, ya que, cuantos más marcos aparezcan en la página, será más difícil desplazarse a través de ella.

Por otro lado, para crear los enlaces a otras páginas se insertan hiperenlaces que lleven a esas páginas determinadas. Cuando inserta un hiperenlace, debe también especificar dónde y cómo va a abrirse el nuevo documento. Si elige que se muestre en un marco determinado, cuando el usuario pulse el enlace, la página se abrirá en el marco especificado.

### 2.3.2. Limitaciones de los marcos

Una de las limitaciones que encuentra a la hora de utilizar marcos es que los robots y arañas de Internet no pueden indexarlos, ya que, se les hace muy difícil leer la información que alberga el marco.

Al principio del desarrollo web era muy frecuente utilizar marcos en las páginas web para disminuir los tiempos de carga, ya que, en aquellos tiempos la banda ancha era realmente lenta. Sin embargo, actualmente aún siguen existiendo webs que utilizan este tipo de marcos. Por ejemplo, es muy frecuente ver una web con un marco izquierdo donde se muestra un menú o una web con un marco superior el cual alberga el logo de la página y una presentación. Es por esta razón que los buscadores más conocidos han comenzado a indexar los contenidos que están comprendidos entre las etiquetas `<framset>` y `</frameset>`. Aunque hay que comentar que aún quedan muchos robots que son incapaces de indexar este tipo de contenido.

Además, cabe destacar que aún hay navegadores que no soportan los frames. Es por ello que, si quiere que una web que contenga frames se visualice en este tipo de navegadores, debe utilizar el elemento `noframes` donde puede incluir un texto alternativo para que se visualice en este tipo de navegadores, si no hace esto su página directamente no se podrá visualizar.

Otro de los problemas que presentan los marcos es que van en contra de la propia estructura del HTML, ya que, el objetivo de este lenguaje no es indicar cómo se representa cada elemento, sino qué es cada elemento. Pues, el cómo se representan los elementos solo dependerá del navegador que esté utilizando. Por ejemplo, si utiliza un navegador descriptivo, puede ver que los textos saldrán más grandes, los párrafos tendrán una mayor separación entre sí, etc. Continuando con el ejemplo, un navegador por voz, hará pausas entre palabras, denotará un énfasis más acentuado en las frases que lleven signos de admiración o interrogación. Todo esto en conjunto es la verdadera esencia del HTML. Es por ello, que los marcos rompen con su esencia, ya que, la lógica de los marcos no indica qué es cada elemento que tiene contenido, sino dónde está cada elemento contenido.

Otra de las imperfecciones que tienen los marcos es que no se pueden entrelazar con diversas matrices de los mismos. Lo primero que se aprende en el desarrollo web es que cada elemento tiene un nombre único que lo identifica y lo distingue de los demás elementos. Este dato es muy importante, ya que, la esencia de la web es que todos los elementos que forman parte de la misma se encuentran entrelazados entre sí, es por ello que web significa red. Por todo ello, para que un elemento sea referenciado por otro elemento, este debe tener un nombre único para saber qué es lo que referencia a qué. Al identificador que se utiliza en el entorno de red para realizar esta acción se le llama URI, lo cual significa Identificador Uniforme de Recursos.

Teniendo en cuenta lo anterior, es por ello que las páginas web se identifican con una variante especial de uri llamado url, lo que significa Localizador Uniforme de Recursos. La url permite especificar dónde está localizado el propio recurso. La url es la dirección que aparece en la barra superior de cualquier navegador cuando busca una página determinada, por ejemplo, <https://www.google.es>.

---

Ahora, que ya sabe lo que es realmente una url y en qué consiste, cabe destacar que los marcos rompen con este esquema. Esto ocurre porque una página con marcos se compone de dos tipos de elementos, por un lado, tiene el documento HTML que contiene la estructura en sí de los marcos, es decir, lo que está contenido dentro del frameset. Por otro lado, tiene los documentos HTML que contienen la estructura de lo que se muestra en cada marco y que están almacenados de manera externa al documento inicial. Eso lleva a la conclusión de que la url de la página con marcos va a ser siempre la del documento que contiene la etiqueta framset y, por lo tanto, si navega a través de esta página se van a ir cargando distintas páginas, pero a su vez la url se va a mantener estática. Es decir, si un usuario comparte, por ejemplo, esa url y otro usuario clica en ella, a este último usuario se le mostrará la página inicial en su estado inicial. Pues, como ya se ha comentado, la url se mantiene fija y, por lo tanto, no es capaz de enlazar el contenido que quería compartir. Esto sucede porque los marcos están imposibilitando el objetivo primordial que tiene una web, es decir, enlazar los documentos. Lo más parecido que puede hacer con los marcos es enlazar con el frameset determinado y seguir dando instrucciones desde ese punto.

Otra opción que tiene es enlazar de forma directa al documento HTML que precise, lo que pasa es que, si ha eliminado de manera intencional los elementos de navegación para insertarlos dentro de un marco separado, encontrará en un punto donde solo se mostrará el documento HTML solicitado, pero no puede volver atrás, puesto que, ya no existen los elementos de navegación.

Cabe destacar también que, los marcos dan problemas con los navegadores. Aunque actualmente los navegadores más conocidos sean capaces de indexar y seguir el contenido de los marcos y todos los documentos que están enlazados a ellos y que forman el inicio de una página, si un usuario, por casualidad, se topa con uno de los documentos que está enlazado a un marco, entrará en él, pero no podrá ver ningún tipo de menú de navegación. Esto sucede porque el usuario no se ha topado con la página principal sino con el documento que debería ir en un marco de la página principal.

Otro de los problemas que se encuentra con los marcos es que solo se puede cambiar el contenido de un marco al mismo tiempo. Ponga un ejemplo donde tiene tres marcos. En el primer marco alberga un índice general de capítulos de un libro. En el segundo marco albergará un subíndice de secciones y en el tercer marco se mostrarán los contenidos de la sección que quiere observar. La lógica de funcionalidad de estos tres marcos es que, al elegir un capítulo del primer marco, cambie el subíndice de secciones del capítulo elegido que se encuentra en el segundo marco y, a su vez, que se muestre el contenido de esa sección en el tercer marco. Sin embargo, esto no es posible, ya que, al elegir un capítulo del índice de capítulos, solo puede cambiar el contenido de uno de los dos marcos restantes.

Por otra parte, los marcos también ponen un alto nivel de complejidad en las páginas. Esto sucede en las páginas que utilizan más de dos marcos y, sobre todo, si quiere que el funcionamiento sea que cambien los contenidos de más de un marco a la vez. Los métodos que se utilizan para realizar esa función los verá más adelante.



TOME NOTA

**Debe tener en cuenta que, debido a los protocolos de red utilizados es más lenta la carga de dos documentos pequeños que se almacenan por separado, que la carga de un documento más grande que contenga, a su vez, ambos documentos pequeños.**

Vea a continuación una lista de más problemas que pueden dar los marcos:

El título del documento puede dejar de ser informativo. Esto sucede porque el título del documento siempre será el del documento que contiene el frameset y no el del documento que se esté mostrando en el momento.

Imposibilidad de almacenar una combinación de marcos en la lista de favoritos. Existen navegadores que sí pueden llevar a cabo esta acción. Sin embargo, hay otros muchos que son incapaces.

Inutilidad de identificadores de fragmento. Se puede llegar al punto en que al apuntar a un identificador de fragmento puede quedarse sin la barra de navegación, por lo tanto, éstos quedan inutilizados.

Si una página tiene más de dos marcos, el proceso de navegación por el historial se hace inservible, ya que, como ha comentado anteriormente la url es fija.

Las soluciones que se plantean para paliar estos problemas no son muy buenas y, a veces, pueden dar más quebraderos de cabeza que resolver el problema en sí.

Por ejemplo, el problema de cambiar los contenidos de más de un marco al mismo tiempo se puede solucionar anidando los frameset. Tomando el ejemplo anterior, en vez de crear tres marcos, tan solo crearía dos. En el primer marco seguiría teniendo el índice de capítulos, en cambio en el segundo marco albergaría un nuevo frameset con otros dos marcos. Uno de esos marcos sería el subíndice de secciones, mientras que en el otro se mostraría el contenido de cada sección. Esta solución es buena en cuanto a que puede apuntar a cualquier capítulo, sin embargo, seguiría con el problema de que no podría apuntar a una sección determinada, ni a una parte de esa sección. Asimismo, seguiría sin solucionar el problema de los buscadores, que hará que la navegación por la página sea más lenta. Esto se debe a que cuando el usuario cambie de capítulo se deberán cargar tres nuevos documentos.

Este método, que ha comentado anteriormente, puede dar más de sí haciendo que cada documento sea verdaderamente un frameset. Por lo tanto, cada vez que quiera abrir un documento, la ventana se vacíe utilizando el atributo target="\_top". Una vez ejecutado este atributo, se volverán a dibujar los marcos en la ventana y se rellenarán con los documentos requeridos. De esta manera, puede cambiar todos los marcos que deseé al mismo tiempo y tendrá más flexibilidad a la hora de apuntar a las diferentes combinaciones de marcos. Sin embargo, esto tiene un inconveniente, y es que, la navegación a través de esta página será mucho más lenta, puesto que, cada vez que clique en un vínculo, la página deberá recargar todo el frameset y todos los documentos que los marcos

alberguen. Y, como pasaba anteriormente, el problema de los buscadores y el de los identificadores de fragmento persiste también en este método. Cabe destacar que con este método no puede dejar fijo ningún marco en la página y, esto es un gran inconveniente para la gente que decide usar marcos para mantener cierta sección de la página estática.

Para rematar este tema, debe saber que existen scripts, creados con JavaScript, que permiten solucionar alguno de estos problemas como, por ejemplo, cargar el frameset de manera automática cuando acceda directamente a uno de los documentos desde un navegador. En cambio, uno de los pilares del diseño web es que cualquier página debería funcionar de manera correcta, aunque el usuario no tenga activado JavaScript en el navegador. Es por ello que, los métodos que ha aprendido son, tan solo, una solución parcial a los problemas, pero no son la solución real.

En conclusión, los marcos a priori parecen una buena idea para implementarlos en su web. Sin embargo, a la larga aportan demasiados problemas y no son tan necesarios como parecen. Por lo tanto, siempre que puede evitar implementar marcos en su página, los evita. Aunque, si va a utilizarlo por cualquier motivo, es importante que utilice las etiquetas noframes y pone información importante sobre su web para que, aquellos usuarios que no puedan utilizar un navegador que soporte frames, puedan acceder a su contenido de alguna otra manera. Estos usuarios pueden ser, por ejemplo, invidentes que necesiten un navegador por voz o, usuarios que no puedan acceder a un equipo actualizado y, por lo tanto, no puedan utilizar la versión más actualizada de un navegador.

### 2.3.3. Alternativas a los marcos

#### Utilización de #include

Como primera alternativa al uso de frames, va a explorar las posibilidades que aporta el uso de #include. Estos elementos son pequeños trozos de código que son comunes a varias páginas, pero que, a su vez, separa en diferentes documentos donde estarán listos para ser utilizados cuando lo necesite.

Este tipo de elementos en HTML son parte de las directivas SSI. Esto quiere decir, que los includes ponen dinamismo a la web sin verse obligado a incluir código PHP, JSP o ASP. Al utilizar los includes, va a poder trabajar de manera independiente en las diversas partes en las que se divide un documento web.

Para explicar este elemento de una manera más profunda, va a tomar como referencia de la división de su página web principal el siguiente ejemplo:

Cabecera	
Cuerpo lado izquierdo	Cuerpo lado derecho
Pie	

Como puede observar, su web se divide en una cabecera, el cuerpo se divide en dos partes y luego, tiene un pie de página. Debe saber que, por ejemplo, partes como la cabecera y el pie son partes que repetirán a lo largo de las diferentes páginas de su web, por lo tanto, si quiere realizar un cambio en una de estas partes, debe realizar dicho cambio en todas las demás páginas.

Con el elemento include adoptar una forma de trabajar como si se tratasesen de documentos HTML independientes. Esto sucede porque al trabajar con este elemento, tan solo necesita insertar las partes mediante una pequeña línea de código. Trabajar con este elemento sería parecido a trabajar con iframes, la única diferencia es que cuando trabaja con iframes el servidor debe procesar el documento incrustado en el mismo.

Para poder utilizar el SSI en sus páginas web, primero debe montar un servidor web en su equipo. Una herramienta para ello puede ser, por ejemplo, un servidor Apache. Esta herramienta viene con PHP y MySQL. Posteriormente, debe habilitar el SSI con la configuración de su servidor Apache. Para ello, accederá al archivo httpd.conf y debe hacer unos cuantos cambios.

El primer cambio que hará será añadir el siguiente código a su archivo:

Options+Includes

Para ello, buscará la etiqueta <Directory> y el código quedará como se muestra a continuación:

```
<Directory>
```

```
Options Indexes FollowSymLinks
```

```
Options+Includes
```

```
AllowOverride All
```

```
</Directory>
```

A continuación, debe incluir en el archivo httpd.conf el siguiente código dentro del apartado que puede ver en el siguiente ejemplo:

```
<IfModule mime_module>:
```

```
AddType text/html .shtml
```

```
AddOutputFilter INCLUDES .shtml
```

Cuando haya realizado estos cambios, debe reiniciar su servidor Apache y, con ello, podría comenzar a utilizar los includes en sus documentos HTML.

Además, para utilizar includes en sus páginas, debe de asegurarse de que estas tienen una extensión .shtml, en lugar de la clásica extensión .html. Sin embargo, los archivos que incluyan dentro de su web podrán tener las extensiones clásicas. El ejemplo quedaría como puede observar a continuación:

<!--#include virtual="header.html"-->	
Cuerpo izquierdo	Cuerpo derecho
<!--#include virtual="footer.html"-->	

Como puede observar en el ejemplo, si realiza su cabecera en un archivo externo, por ejemplo, header.html y lo pone a su página principal index.shtml, tan solo debe escribir la siguiente línea de código:

```
<!--#include virtual="header.html"-->
```

De este modo, puede trabajar de manera separada las partes de una página. Esto conllevará a que tenga una estructura más sólida, ya que, si tiene el archivo header.html añadido en más de un documento y quiere modificar el header, tan solo tendrá que modificarlo en su propio archivo header.html y se modificará en todos los archivos.

### Diseño con tablas

Llegados a este punto, con los elementos que ha aprendido junto con las propiedades que aporta el código CSS, capaz de distribuir el contenido de su web de muchas maneras posibles. Sin embargo, aún queda un punto que explorar, y es que, si quiere distribuir los elementos en forma de columnas y filas tabuladas aún no puede hacerlo. Para ello, precisamente, debe echar mano del diseño con tablas.

Como ya sabrá, la estructura de las tablas se distribuye en una o más filas que, a su vez, se dividen en una o más columnas. Esta división entre filas y columnas crea un elemento más pequeño al que denomina celdilla, que es donde va añadida la información que contiene la tabla.

Si mira la tabla como una estructura lógica, puede tener un encabezado, un cuerpo y un pie al igual que las páginas web. Y, en las celdillas es donde se almacenarían todos los tipos de datos, como textos, imágenes, títulos, etc. Además, cada celdilla puede tener su propio tamaño, habrá celdillas que ocupen más filas o columnas que las demás. Es por ello que, las tablas son una estructura muy flexible, por eso, se utilizan muy a menudo como parte fundamental de los documentos, ya que, con ellas puede distribuir diferentes partes controlando de manera eficaz las dimensiones y las posiciones de los elementos. Sin embargo, las tablas también plantean algunos problemas, puesto que, pueden causar errores en algunos tipos de navegadores. Sobre todo, en los navegadores visuales. Es por ello que, el código CSS aporta una gran solución a este problema, pues es capaz de ofrecer el mismo control que el que ofrecen las tablas y, además, hace que los documentos sean legibles para todo tipo de navegadores.

Crear una tabla en HTML es un trabajo bastante complejo, puesto que, para su creación intervienen un gran número de elementos con sus respectivos atributos. En HTML, por ejemplo, no podría insertar una celdilla de manera directa en la tabla, antes debería crear la estructura determinada, que en este caso sería una fila, para posteriormente insertar la celdilla. Además, las filas se pueden agrupar como estructuras que forman parte del header, footer o del propio cuerpo de la tabla. Lo mismo, puede hacer con las columnas. Vea el siguiente ejemplo:

Título		
Datos	Datos	Datos
Pie		

Como puede observar, esta imagen muestra una tabla que es un elemento donde están añadidos los demás elementos más pequeños. Lo primero que puede ver en la tabla es el título, el cual él mismo es otro elemento. Debe tener en cuenta que los títulos y los datos son el contenido de los elementos. A continuación, puede ver el grupo de filas de datos, el del pie y el del título que son elementos mayores si los compara con una sola celdilla. Y, por último, en cada fila también hay elementos. Por otro lado, los bordes, fondos y márgenes son atributos de la tabla.

Tal como ha visto en el ejemplo anterior, toda la tabla en sí es un elemento llena, a su vez de pequeños elementos. Es decir, en el nivel más alto estaría la tabla como elemento principal, y en su interior, puede ver más elementos pequeños como grupos de fila, grupos de columna, títulos, datos, etc. Puede asemejarlo al cuerpo humano donde el cuerpo es el elemento principal y dentro de él puede encontrar múltiples sistemas de órganos, a su vez, órganos individuales y así hasta llegar a la célula como unidad.

Resumiendo, table es el elemento principal y, por lo tanto, el elemento que se definirá primero en su documento HTML. Después, incluye en el elemento principal, elementos tales como filas, columnas, títulos, imágenes, texto. Al ser la tabla un elemento de bloque, no puede insertarla en mitad de un párrafo de texto, ya que, provocaría una interrupción del elemento párrafo. Sin embargo, la puede introducir dentro de un div. Para crear una tabla, debe introducir tanto la etiqueta de apertura `<table>`, como la etiqueta de cierre `</table>` siendo ambas totalmente obligatorias. A continuación, va a ver una serie de atributos que puede utilizar con el elemento table:

- **Id.** Este atributo asigna un nombre identificador al elemento, como ya sabe de veces anteriores, el identificado debe ser un nombre único para todo el documento. Esto se utiliza para referenciar a este elemento desde, por ejemplo, un script.
- **Class.** Este elemento asigna un nombre de clase al elemento. El atributo class se suele utilizar para añadir estilos CSS al elemento. Gracias a este atributo puede utilizar un estilo para un grupo de elementos.
- **Style.** Este atributo es utilizado para incluir código CSS directamente en el elemento. Aunque esto pueda resultar útil en algunos casos muy específicos, cuando inserta código CSS es mejor hacerlo desde un archivo externo y relacionar el código CSS con el elemento mediante el nombre de clase. Esta es la práctica más profesional y eficiente.

- **Title.** La función de este atributo es asignar un título al elemento. Dentro de este atributo debe insertar una descripción corta sobre el elemento al que le aplica el título. Normalmente, el contenido de este atributo se muestra en algunos navegadores al poner el puntero del ratón encima del elemento.
- **Lang.** Este atributo se utiliza para especificar el idioma de lo que hay dentro del elemento. El valor de este atributo por defecto es desconocido. Sin embargo, si escribe en un documento XHTML, el atributo que utiliza para especificar el lenguaje será xml:Lang. Ambos tipos de estándares son compatibles, así que, puede utilizar estos dos atributos indistintamente. Lo único que debe tener en cuenta es que en documentos XHTML 1.1, el atributo Lang ya no es válido y debe utilizar, en su lugar, el atributo xhtml:Lang. Este atributo indica la dirección de escritura del texto que contiene el elemento. Este atributo tiene dos opciones:
  - RTL: lectura de derecha a izquierda.
  - LTR: lectura de izquierda a derecha.
- **Summary.** Este atributo aporta un resumen de la estructura de la tabla en general. Lo que hace la información que alberga este atributo es expandir el contenido del elemento caption de HTML.  
El formato de los resúmenes que debe introducir en este atributo es que sea corto, preciso y lo más descriptivo que sea posible. Esto se debe a que estos resúmenes son muy importantes para ayudar a que los usuarios invidentes puedan interpretar el contenido de una tabla.
- **Width.** Este atributo se utiliza para dar un ancho específico a una tabla. Si no especifica un ancho determinado, serán los navegadores quienes se encargarán de dárselo.
- **Frame.** Este atributo se utiliza para especificar qué bordes de alrededor de la tabla se mostrarán. A continuación, verá una lista de opciones posibles para determinar este atributo:
  - **Border:** se muestran todos los lados.
  - **Box:** Se muestran todos los lados.
  - **Rhs:** solo se muestra el lado derecho.
  - **Lhs:** solo se muestra el lado izquierdo.
  - **Vsides:** solo se muestran el lado izquierdo y el derecho.
  - **Hsides:** solo se muestran el lado superior y el inferior.
  - **Below:** solo se muestra el lado inferior.
  - **Above:** solo se muestra el lado superior.
  - **Void:** No se muestra ningún lado. Este es el valor por defecto.
- **Rules.** Este atributo especifica qué reglas se deben mostrar entre las celdillas de una tabla.

Vea una lista con las opciones de las que dispone:

- **All:** Se muestran las reglas entre todas las filas y todas las columnas.
- **Cols:** Solo se muestran las reglas entre grupos de columnas.
- **Rows:** Solo se muestran las reglas entre grupos de filas.
- **Groups:** Se muestran las reglas entre grupos de filas y columnas.
- **None:** No se muestran ninguna regla. Este es el valor por defecto.

- **Border.** Este atributo especifica el ancho del borde de una tabla. Su valor se escribe en píxeles.
- **Cellspacing.** Este atributo especifica el ancho del espacio que se dejará entre las celdillas. También, se utiliza para especificar el espacio entre el borde de la tabla y las celdillas continuas.
- **Cellpadding.** Este atributo especifica el ancho del espacio que se va a dejar entre el contenido de la celdilla y sus propios bordes.
- **Align.** Este atributo especifica una alineación horizontal de la tabla.

Vea a continuación una lista de las posibles opciones que puede utilizar con este atributo:

- Center: con esta opción centra la tabla.
- Left: con esta opción alinea la tabla hacia la izquierda.
- Right: con esta opción alinea la tabla hacia la derecha.
- Debe saber que este atributo, en las versiones actuales de HTML, está obsoleto y, por lo tanto, no se recomienda su uso.
- El elemento table, a su vez, está compuesto por otros elementos los cuales va a ver a continuación en la siguiente lista:
- Tbody: Este elemento es de uso obligatorio y aparece una o más veces agrupando las filas que componen el cuerpo de la tabla.
- Tfoot: Este elemento es de uso opcional, sin embargo, es un elemento que no puede repetirse. Este elemento permite agrupar las filas que se encuentran en el pie de la tabla.
- Thead: Este elemento, también, es de uso opcional y, al igual que el elemento anterior, no puede repetirse. Este elemento permite agrupar las filas de la cabecera de la tabla.
- Col/colgroup: Este elemento es un elemento opcional también y tiene como finalidad introducir características a las columnas de la tabla. Este elemento, a diferencia del anterior, puede repetirse.
- Caption: Este elemento no puede repetirse y puede utilizarlo de manera opcional. Este elemento tiene como finalidad contener el título de la tabla.

De todos los atributos que ha visto que se pueden utilizar con el elemento table, uno de los más importantes es el atributo summary. Esto se debe a que su finalidad es aportar a los usuarios invidentes un resumen sobre el contenido de la tabla. Este resumen puede ser leído por un sintetizador de voz o ser utilizado por un dispositivo en braille. En estos casos, la información comenzará a transmitirse por este resumen y continuará transmitiéndose enumerando el contenido que aparece en cada fila y columna de la tabla.

Por otro lado, el color de fondo de la tabla lo puede definir con el atributo bgcolor al que puede asignar tanto el nombre de un color, como el código hexadecimal del mismo.



TOME NOTA

**Como alternativa al uso del atributo bgcolor, puede crear una hoja de estilo donde utilice la propiedad background-color aplicándola a través de un selector a una tabla. Este es el método que se recomienda emplear para establecer el color de fondo, ya que, bgcolor es un atributo obsoleto.**

## Utilización de bloques

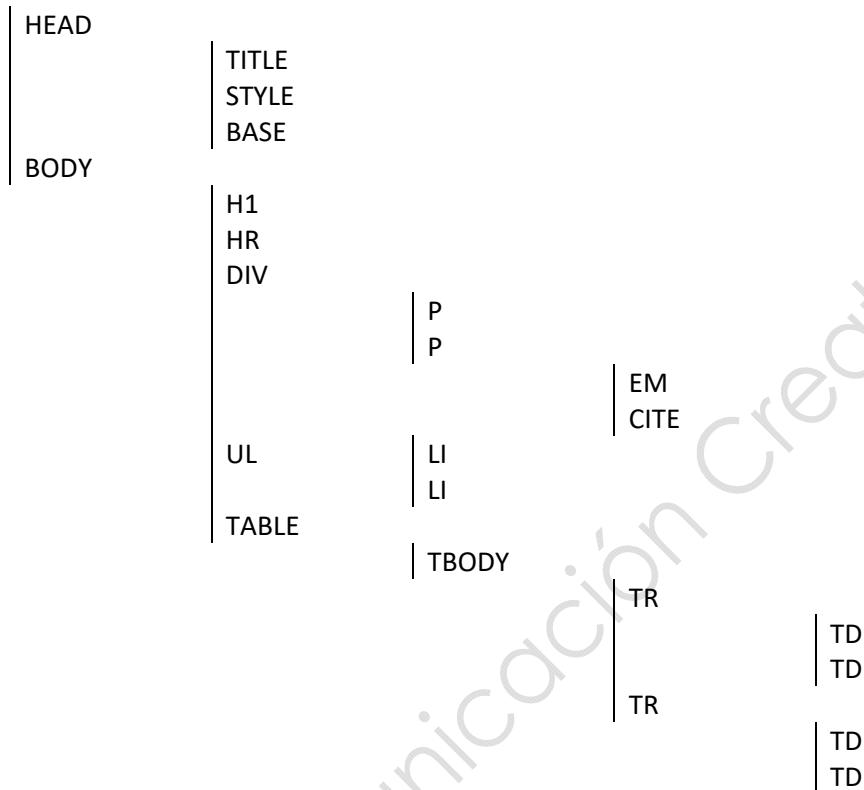
Ahora, ya conoce dos maneras en las que distribuir el contenido de su web creando distintas secciones con una posición y dimensiones fijas y determinadas. Para ello, ha aprendido a crear tablas y marcos, pero como ha ya visto, estas dos opciones aportan más problemas que beneficios. Esto se debe a que, por ejemplo, las tablas no están diseñadas para funcionar como esqueleto estructural de una página, sino para albergar conjuntos de datos en su interior. Por otra parte, los marcos son elementos que con el tiempo irán desapareciendo, ya que, los navegadores no visuales lo tienen muy difícil para visualizar su contenido, por lo tanto, los marcos no aportan utilidad ninguna. Por lo tanto, como alternativa queda implementar código CSS para crear las diferentes secciones de su página web y, de ese modo, estructurar la información dentro de ella. Para ello, con código CSS establece distintos tamaños y zonas. Además, con CSS3 cuenta con nuevas propiedades que ayudarán a conseguir su objetivo de manera más sencilla. De hecho, anteriormente ha aprendido alguna de estas propiedades, las cuales son compatibles con los navegadores más importantes y que más usuarios usan. Por ejemplo, navegadores como Chrome, Mozilla Firefox, Edge, etc. Serán capaces de interpretar las instrucciones que escriba con CSS3.

Al utilizar código CSS, puede controlar totalmente la distribución del documento que esté creando empleando, por ejemplo, instrucciones como float, clear, width, etc. En conjunto con otras nuevas propiedades como position, left, top, con las que puede ajustar totalmente las posiciones de los elementos de su web y las dimensiones de cada zona donde se albergan los mismo. De este modo, puede conseguir divisiones de zonas como las que vio que podía crear con los propios marcos, pero sin el problema de tener que dividir el contenido entre varias páginas para que funcionasen al unísono y, sin la necesidad de utilizar propiedades obsoletas.

Por otra parte, es de mera importancia conocer de qué manera trabajan los navegadores en el tratamiento de los elementos en línea, en bloque y cómo estos generan las zonas por las que se distribuye el contenido. Puede estructurar un documento HTML en forma de árbol haciendo énfasis en la jerarquía que existe entre los elementos del mismo. Si quiere entender cómo los navegadores, sobre todo los visuales, estructuran los diferentes elementos en sus pertinentes bloques, necesita tener en cuenta la relación jerárquica de cada uno de los elementos en relación a los demás. En este proceso lo verdaderamente importante no es el contenido útil de los propios elementos, sino la relación existente entre ellos mismos. En el caso de realizar una extracción de los elementos, los cuales podría reconocer por sus etiquetas de apertura, y los situase en un esquema de árbol donde el tronco principal sería el elemento HTML de donde derivarían los elementos head y body y,

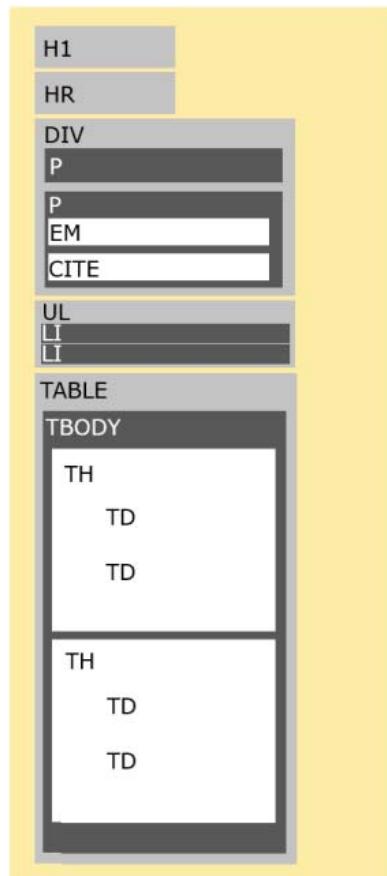
además, de estas dos ramas derivarían todo el resto de elementos del documento. Como resultado obtendría un esquema como el siguiente:

HTML



Si en el árbol de elementos elimina aquellos que no aportan contenido a la página, tendría que eliminar head entero y solo quedaría con la rama de body, que es donde están contenidos todos los elementos que aportan algún contenido a la web.

Si quisiera representar la parte del body de este esquema de árbol como esquema de bloques, debería introducir cada elemento en su propio recuadro y ese recuadro en el bloque correspondiente. Vea el ejemplo a continuación:



Como puede observar en la imagen, cada elemento sin importar su tipo, está dentro de un bloque de visualización. Dentro de estos bloques es donde se introduce el contenido propio del elemento y sus márgenes determinados. El contenido que albergan estos bloques pueden ser texto y otros elementos.

Si se fija en el esquema, puede observar el elemento de mayor nivel jerárquico, el cual actúa como contenedor de los demás elementos. El elemento de mayor nivel es el **body** que alberga cinco bloques en su interior. Puede observar que se albergan el **h1**, **hr**, **div**, **ul** y **table**. Los bloques, **h1** y **hr**, no albergan más bloques en su interior. Por lo tanto, dependiendo de cómo haya configurado su contenido, ambos elementos podrán contener bordes, un fondo, márgenes, etc. Se mostrará en el documento final. En cambio, el bloque del **div** sí contiene dos bloques más en su interior. Estos dos bloques tendrán su propia configuración específica, además de la configuración que heredan del bloque **div** y del bloque **body**.

Por último, los elementos del último nivel, los cuales son **em**, **cite** y **td**, están también albergados en un bloque propio y, también, heredan las propiedades de los bloques que tienen niveles superiores. Sin embargo, la distribución de estos bloques es diferente a la de los demás, cosa que verá más adelante. Las propiedades que pueden contener los elementos pueden clasificarse en cuatro grupos:

Margen interior

Margen exterior

### Borde del bloque

### Fondo del bloque

Si quiere pasar a visualizar el documento, lo primero que hará el navegador es examinar el código del mismo, a continuación, analizará el código y, posteriormente, creará un árbol de elemento como en el ejemplo que vio anteriormente. De este modo, el navegador podrá calcular el tamaño del documento. Que el navegador tome estos datos es fundamental para poder posicionar los diferentes bloques, con su respectivo contenido, en las zonas de visualización disponibles en la ventana del navegador. El navegador hace el cálculo de las dimensiones del bloque partiendo de la base del contenido de la misma. Por ejemplo, si el bloque contiene un texto con determinada fuente, el navegador calculará el espacio específico para esa fuente determinada. En cambio, si en vez de un texto tiene una imagen, el navegador calcularía las dimensiones para las dimensiones originales de la imagen insertada. Resumidamente, si el bloque tiene un tamaño determinado y dentro de él hay diferentes elementos, habrá que sumarle al bloque el tamaño suyo y el de los elementos contenidos en él. Es de esta manera como el navegador hace un cálculo total de las dimensiones. Además, también debe tener en cuenta que al espacio que calcula el navegador, debe sumarle el espacio que ocupan los márgenes interiores y exteriores de cada uno de los elementos. El resultado de todas estas sumas será el tamaño resultante del bloque. De este modo es como el navegador aporta un cálculo por defecto, ya que, mediante código CSS puede variar la altura y el ancho fijo de los elementos con las propiedades width y height. Aunque, hacer esto no es lo habitual, a no ser que le indique al elemento al que le ha aplicado las propiedades en qué posición del bloque debe aparecer, así hará que los demás elementos fluyan a su alrededor.

En el momento en que se conoce certeramente las dimensiones de las cajas, el navegador podrá ir posicionándolas por la ventana según las instrucciones recibidas. Los elementos que encontrará el navegador pueden ser de línea o de bloque, aunque existen más tipos de elementos, estos dos son los más frecuentes.

El navegador recorrerá los bloques según la jerarquía que tengan los bloques en el árbol y, dependiendo del tipo de bloque, los posicionará de manera vertical u horizontal. Si, por ejemplo, el navegador se encuentra un elemento de bloque como h1, div o p, pasará a leer los elementos que tengan por debajo. Es decir, que leerá la siguiente ventana de visualización o la línea siguiente del bloque contenedor. Si, por el contrario, se topa con un elemento de línea como em o cite, la posicionará a continuación del bloque anterior y en la misma línea de visualización. En el caso en que se llegue al límite, el margen derecho del bloque contenedor, se posicionará en la línea siguiente, sin embargo, este bloque contenedor conformaría una sola unidad. Los bloques pertenecientes a los elementos cite y em, se posicionarán a lo ancho de el bloque del elemento p, en el cual em y cite están contenidos. El posicionamiento de los bloques de los elementos dependerá del valor de la propiedad display de cada uno de los propios elementos. Puede modificar el valor de esta propiedad al igual que modifica cualquier propiedad. Los valores que puede tomar la propiedad display son inline y block, que corresponden a elementos en línea y elementos en bloque respectivamente.

Vea, a continuación, una lista con las propiedades que puede utilizar con los elementos:

- **Inline-block.** El elemento genera un bloque block que se comporta como si fuera un bloque inline pudiendo añadir más elementos en la misma línea.
- **List-item.** El elemento se comporta como si fuese un elemento li.
- **None.** No se muestra el elemento. Esta opción hace como si el elemento no existiera y, por lo tanto, otros elementos podrán ocupar su espacio.
- **Inline.** El elemento se muestra en un bloque en línea.
- **Block.** El elemento se muestra en un bloque en bloque.

Vea la lista de propiedades que hacen que un elemento simule el comportamiento de otro elemento:

Table-row-group

Table-row

Table-header-group

Table-footer-group

Table-column-group

Table-column

Table-cell

Table-caption

Table

Inline-table

Lista de otras propiedades más avanzadas:

Run-in

Inline-grid

Grid

Inline-flex

Flex

- **Inherit.** Se heredan las características del elemento padre.

Todos los navegadores disponen de una hoja de estilo por defecto donde se establece el tipo de bloque que corresponde a cada elemento del HTML. Sin embargo, puede modificar esta

configuración como cualquiera de las demás. Para ello, utiliza selectores específicos en su hoja de estilo. Vea un ejemplo:

Li {display: none}

P {display: inline}

Si pone estas dos líneas al elemento style, va a hacer que los elementos li no tengan bloque de visualización, es decir, que no serán visibles en el documento. En cambio, los elementos p sí tendrán bloques en línea y se posicionará horizontalmente en vez de en manera vertical.



TOME NOTA

***Debe conocer que la propiedad display aparece en CSS1, pero algunos de sus valores son exclusivos de CSS3, por lo que, no se recomienda modificar el tipo de bloque asociado a los elementos predefinidos en HTML***

## 2.4. Trabajar con ventanas

Puede abrir ventanas independientes en el navegador de tres maneras diferentes.

- La primera es con el código siguiente:

```
Pulse aquí
```

- La segunda manera de abrir una nueva ventana en el navegador es utilizar código JavaScript, así, puede abrir la ventana con las opciones y dimensiones que quiera.
- La tercera opción es utilizar JavaScript de manera que abra automáticamente una nueva ventana con las opciones y dimensiones deseadas.

Para utilizar JavaScript, debe insertar el código siguiente entre las etiquetas head del documento HTML principal. Véalo a continuación:

```
<script language="JavaScript">
```

```
nuevaVentana=window.open("principal.htm","","","width=xxx, height=xxx")
```

```
nuevaVentana.creator=self
```

```
</script>
```

Principal.htm será el nombre de la nueva ventana independiente que se abrirá si aplica este código, su anchura y altura se establecerán con las propiedades height y width. Por ejemplo, podría establecer los siguientes valores height=165 y width=317.

Cuando haya finalizado de crear este código, solo tendrá que crear el nuevo archivo para la nueva ventana independiente.

#### **Utilizar el elemento div:**

Principalmente, necesita un elemento donde albergar el contenido de la web que desea mostrar. Para ello, utiliza la etiqueta div. Esta etiqueta creará una especie de contenedor que será parte de la página, pero no se mostrará cuando se abra la página, es decir, este elemento será invisible. El contenedor solo se mostrará cuando tenga elementos dentro para mostrar. Además, también necesita una manera de ocultar los elementos cuando desee. Vea el siguiente código:

```
<div id="miNuevaVentana" style="position:fixed; width:350px; height: 190px; top:0; left:0; Font-family:Verdana,Arial,Helvetica,sans-serif; Font-size:12px; font-weight: normal; border:rgba(206, 27, 27, 0.767) 3px solid; background-color: #52a310; color:#0000; display:none;">
```

Si pone este trozo de código en su archivo HTML, verá que este contenedor no se va a visualizar en el navegador. Esto se debe a que la propiedad display:none hace que el elemento sea invisible.

Para resumir, la etiqueta div permite crear un contenedor para añadir elementos. Las propiedades más importantes de div son:

- Id que es el identificador con el cual puede recuperar el elemento para acceder a él desde el DOM.
- Display que es la propiedad con la cual indica al navegador si quiere que muestre el elemento o no.
- Dentro de div es donde añade todos los elementos, pero debe procurar que esos elementos quepan dentro del div.

Lo que hará a continuación es crear un elemento en JavaScript que insertará en su documento HTML. Véalo a continuación:

```
Function mostrarVentana ()
```

```
{
```

```
Var ventana = document.getElementById('miNuevaVentana');
```

```
Ventana.style.marginTop="100px";
```

```
Ventana.style.left=((document.body.clientWidth-350)/2)+"px";
```

```
Ventana.style.display='block';
}

Con esta función JavaScript referencia al contenedor, que crea anteriormente, mediante su id. Además, coloca el contenedor en el centro del documento y lo hace visible.
```

Para finalizar, creará la función que referenciará al contenedor mediante su id y lo ocultará haciendo que sea totalmente invisible cuando visualice la página. Vea el siguiente ejemplo:

```
Function ocultarVentana ()
{
Var ventana=document.getelementById('miNuevaVentana');
Ventana.style.display='none';
}
```

#### **2.4.1. Creación de varias ventanas**

En este apartado, va a aprender a crear ventanas. Lo primero que hará es crear un código con JavaScript capaz de abrir varias ventanas con clicar una sola vez. Va a ver un ejemplo donde tendrá las variables x e y, las cuales van albergar la información de dos ventanas como, por ejemplo, posición, medidas, url de la ventana que hay q cargar, etc. En este ejemplo, tan solo creará dos ventanas, pero puede crear tantas como quiera. Véalo a continuación:

JavaScript:

```
<script language="javascript">

Function abrirVentanas()
{
X>window.open("http://www.abriendoprimeraventana.com","nombrePrimeraVentana",
"width=385, height=180, top=0, left=0,status,toolbar=1, scrollbars, location");

Y>window.open("http://www.abriendosegundaventana.com","nombreSegundaVentana",
"width=385, height=180, top=0, left=0,status, toolbar=1, scrollbars, location");
}
```

}

</script>

<p><a href="javascript:abrirVentanas()">Pulse aquí</a>

A continuación, va ver cómo realizar el código que sea capaz de abrir varias páginas desde un solo enlace. Para ello, va a hacerlo de dos maneras. La primera manera es con una función. Vea el ejemplo siguiente:

```
<script language="javascript">
```

```
Function abrirPaginas(){
```

```
Window.open('http://url1.com','_blank');
```

```
Window.open('http://url2.com','_blank');
```

```
//Si quiere abrir más páginas a la vez, repetirá el mismo código con diferentes url.
```

```
}
```

```
</script>
```

Para agregar otras páginas volverá a escribir el siguiente código:

```
Window.open('http://url','_blank');
```

Después, al enlace le debe añadir la función que crea anteriormente. Vea el ejemplo:

```
Imagen o texto
```

La segunda manera que tiene para hacer el mismo proceso, puede verla en el siguiente ejemplo:

```
<a href="javascript:void(0);" onclick="javascript:window.open('http://url1','_blank');
```

```
Window.open('http://url2','_blank');">Imagen o texto
```

Para agregar más páginas utiliza la siguiente línea de código:

```
Window.open('http://url','_blank');
```

### **Abrir enlaces de una web en una nueva ventana**

Para hacer que los enlaces se abran de manera forzada en una nueva ventana en cualquier navegador, debe utilizar JavaScript para hacer cumplir unas especificaciones determinadas en XHTML para que, de ese modo, su página se valide correctamente.

Es muy frecuente encontrar webs donde se necesite utilizar este tipo de enlaces que se abren en una nueva ventana, ya que, si al clicar en el enlace el usuario es redirigido a esa otra web, lo que consigue es que el usuario salga de su web y, eso, no interesa. Así, si obliga a ese enlace a abrirse en una nueva ventana, le ofrece al usuario la opción de visitar ese nuevo enlace, pero sin perder contacto con su web.

Sin embargo, con la versión actual de XHTML no es fácil realizar este tipo de validación, ya que, antiguamente sí era efectivo hacer que los enlaces se abrieran con target="\_blank", pero la versión actual de XHTML no permite la validación al utilizar este atributo. La manera que tiene de solucionar este problema es utilizar código JavaScript, puesto que, todos los navegadores soportan este lenguaje y, además, viene habilitado por defecto.

#### **Abrir enlaces en una nueva pestaña del navegador**

El navegador Internet Explorer siempre ha tenido problemas para poder abrir los enlaces en una nueva pestaña, sin embargo, desde su versión 10, la cual venía incluida en el sistema operativo Windows 8, incluye la posibilidad de abrir nuevos enlaces en nuevas pestañas del navegador. Además, una nueva versión HTML5 es capaz de validar y dar soporte al atributo target="\_blank" al agregar enlaces en páginas web. Vea un ejemplo de cómo emplear HTML5 para abrir una nueva pestaña en el navegador Internet Explorer:

```
Nombre del enlace
```

#### **Abrir nuevas ventanas en el navegador utilizando JavaScript**

Crear un enlace que se abra en una nueva pestaña es algo sencillo de hacer en los navegadores más modernos. Para ello, debe utilizar código JavaScript, con el cual creará una función utilizando la función window.open() de la que verá su sintaxis en el siguiente ejemplo:

```
Window.open(url,nombre,opciones)
```

A partir de su sintaxis, puede observar que donde pone url inserta la url del enlace. El siguiente parámetro es el nombre, el cual es opcional y, por último, puede insertar opciones que permiten especificar las propiedades que utilizará su ventana al abrirse.

A continuación, va a ver un código con el que creará un vínculo que abrirá una nueva ventana en la esquina superior izquierda de la pantalla. Esta ventana tendrá una separación de 50px en cada borde, además, sus dimensiones serán 400px de altura y 600px de anchura. Vea el código a continuación:

```
Ventana nueva

```

Ahora, va a crear un botón que al pulsarlo abra la nueva ventana. Vea el siguiente ejemplo:

```
<input type="button" value="Ventana nueva" onclick="javascript:window.open('http://url.com','','width=600,height=400,left=50,top=50,toolbar=yes');"/>
```

Por último, va a ver una lista de opciones que puede utilizar para crear este tipo de ventanas con JavaScript:

- **Location.** Se utiliza para permitir mostrar la barra de direcciones, sus opciones son yes, o no.
- **Menubar.** Muestra o no la barra de menú. Opciones a utilizar yes o no.
- **Toolbar.** Muestra o no la barra de herramientas. Opciones a utilizar yes o no.
- **Scrollbars.** Pone o no barras deslizantes. Las opciones a utilizar son true, false, yes o no.
- **Resizable.** Permite cambiar el tamaño de la ventana. Opciones yes, no, true o false.
- **Top.** Posiciona la ventana y permite añadir una distancia en píxeles del borde superior de la ventana.
- **Left.** Posiciona la ventana y permite añadir distancia en píxeles entre la ventana y el borde izquierdo.
- **Width.** Pone un ancho en píxeles o porcentaje a la ventana.
- **Height.** Pone una altura en píxeles o porcentaje a la ventana.
- **Directories.** Permite mostrar la barra de comandos. Opciones yes o no.
- **Status.** Muestra la barra de estado. Opciones yes o no.

#### 2.4.2. Interactividad entre varias ventanas

##### Método open()

Como ha visto anteriormente el método open es un objeto de window que se utiliza para abrir ventanas. Su sintaxis es la siguiente:

```
Window.open()
```

Si ejecuta esta función en este momento, el navegador abrirá una ventana vacía, puesto que, no le ha indicado ningún documento a abrir en la función. En cambio, si quiere que esta función abra un documento deseado debe aportarle una url como parámetro. Vea el ejemplo:

```
Window.open('paginaejemplo.html')
```

Ahora, esta función abrirá el documento paginaejemplo.html y la ventana mostrará el contenido del mismo.

Puede establecer hasta tres parámetros en el método open, los cuales ninguno es obligatorio. Vea el ejemplo siguiente:

Window.open(documento, nombre de ventana, atributos de ventana)

En el parámetro del documento, debe añadir la uri del documento que quiere que se cargue dentro de la ventana.

Por otra parte, el nombre de la ventana será el identificador de la ventana a abrir. Este identificador se utiliza para tener una referencia de la ventana cuando esté abierta.

Por último, los atributos de la ventana serán una serie de opciones que van a permitir añadirlle a la nueva ventana características como barras que se muestran, posición de la ventana, tamaño de la misma, etc.

Vea a continuación una lista con algunos de los atributos que puede utilizar:

- Directories: muestra o no la barra personal. Opciones yes o no.
- Location: muestra o no la barra de dirección. Opciones yes o no.
- Status: Muestra o no la barra de estado. Opciones yes o no.
- Toolbar: Muestra o no la barra de herramientas del navegador. Opciones yes o no. Debe tener en cuenta que el navegador Internet Explorer versión 7 ignora esta funcionalidad.
- Left: Permite posicionar la ventana desde la izquierda en píxeles.
- Top: Permite posicionar la ventana desde la parte superior en píxeles.
- Height: Pone una altura a la ventana en píxeles.
- Width: Pone una anchura a la ventana en píxeles.
- Scrollbars: Muestra o no la barra de desplazamiento. Opciones yes o no.
- Menubar: Muestra o no la barra de menú. Opciones yes o no.

### Abrir ventanas ante un evento

En el caso en el que quiera que se abra una ventana a partir de un evento, por ejemplo, al clicar un botón, debe definir este evento en el propio manejador de eventos del botón. Vea el ejemplo:

```
Onclick="window.open('ventana.html','ventanaNueva','width=300,height=400')"

<input type="button" value="Pulse aquí"

onclick="window.open('ventana.html','ventanaNueva','width=300,height=400')"/>
```

### Abrir ventanas desde una función

También puede utilizar una función que sea la encargada de abrir una ventana y que, a su vez, sea llamada desde el evento que ha creado en el botón. Vea el ejemplo:

```
<script type="text/javascript">
```

```
Function ventanaNueva(){
```

```
Window.open('ventana.html','ventanaNueva','width=300, height=400')

}

</script>
```

A continuación, creará la llamada a la función dentro del botón que crea anteriormente. Vea el ejemplo.

```
<input type="button" value="Pulse aquí" onclick="ventanaNueva()"/>
```

#### **Usar una misma función para abrir varios documentos**

Para utilizar una única función que permita abrir varios documentos, debe añadir una variable a la propia función, la cual pasará como argumento cuando llame a dicha función. Vea el ejemplo:

```
Function ventanaNueva (documento){

Window.open(documento, 'ventanaNueva','width=300, height=400');

}
```

Por último, creará un botón para cada documento a abrir donde le pasará por parámetro el documento que quiere que abran. Vea el ejemplo:

Botón 1

```
<input type="button" value="Pulse aquí 1" onclick="ventanaNueva ('pagina1.html')"/>
```

Botón 2

```
<input type="button" value="Pulse aquí 2" onclick="ventanaNueva('pagina2.html')"/>
```

#### **Usar una misma función para abrir varios documentos con diferentes tamaños**

También, puede abrir varias ventanas con una misma función donde cada botón aplique unos tamaños diferentes a cada ventana. Para ello, debe establecer los valores de las dimensiones con las que desee que se abran las distintas ventanas, para ello, tiene que pasar como parámetro a la función los valores del alto y el ancho de la ventana. Vea el ejemplo:

```
Function ventanaNueva(documento, ancho, alto){

Window.open(documento,'ventanaNueva','width='+ancho+',height='+alto);

}
```

Ahora, llamará a la función desde los respectivos botones de cada ventana a abrir y establezca las dimensiones. Vea el ejemplo:

Botón 1

```
<input type="button" value="Pulse aquí 1" onclick="ventanaNueva ('pagina1.html', 300, 400)"/>
```

Botón 2

```
<input type="button" value="Pulse aquí 2" onclick="ventanaNueva ('pagina2.html', 420, 550)"/>
```

### Referenciar a la ventana

Puede emplear JavaScript para modificar ventanas que ya están abiertas, pero si quiere realizar esto, antes debe referenciar a la ventana. El segundo parámetro del método open sirve para este cometido, ya que, en él definirá el nombre de la ventana con el cual puede referenciarla mediante JavaScript. Tomando el ejemplo anterior, su ventana se llamaría ventanaNueva, aunque podría haber empleado cualquier otro nombre. Esto es algo opcional como ya vio anteriormente.

En el caso de que quiera hacer algo con su ventana ya abierta como, por ejemplo, cerrarla mediante un botón. Debe invocarla con el nombre que haya asignado en el parámetro de la función open. Vea el siguiente ejemplo:

```
Onclik="ventanaNueva.close()"
```

### Abrir ventanas diferentes desde la misma función

Si quiere que cada ventana que abra tenga su propio nombre identificador, debe añadir una variable más que pasa como parámetro a su función. De esta manera, al abrir distintas ventanas con distintos botones, se abrirán con sus respectivos nombres y no como pasaba en los anteriores ejemplos. A continuación, vea el ejemplo:

```
Function ventanaNueva(documento, ancho, alto,nombreVentana){
```

```
Window.open(documento,nombreVentana,'width='+ancho9',height='+alto);
```

```
}
```

Botón 1

```
<input type="button" value="Pulse aquí 1" onclick="ventanaNueva ('pagina1.html', 300, 400,'ventana1')"/>
```

Botón 2

```
<input type="button" value="Pulse aquí 2" onclick="ventanaNueva ('pagina2.html', 420, 550,'ventana2')"/>
```

Añadiendo ese último parámetro permite que cada ventana que abra lo haga en ventanas diferentes. Por el contrario, en los ejemplos anteriores, al abrir las diferentes ventanas se abrían siempre en la misma ventana, ya que, la referencia de la ventana era la misma para todos los botones. Esto no permitía que dos documentos estuvieran abiertos a la misma vez, ya que, cuando se seleccionaba otro botón la nueva información reemplazaba a la anterior. Sin embargo, al asignar nombres diferentes para cada ventana, puede abrir varios documentos al mismo tiempo.

### Usar enlaces para abrir ventanas

Nunca es recomendable utilizar JavaScript para todo el proceso de acceder a la información y recursos de una web. En vez de eso, puede utilizar un enlace que lleve al documento que deseé mostrar en su ventana. De esta manera, se asegura de que esta información se visualice por parte del usuario, aunque este no tenga soporte para JavaScript. Así, el usuario podrá utilizar este enlace de manera alternativa para llamar al documento. La manera más fácil de hacer esto es utilizando la siguiente función:

```
Enlace a la página
```

Para utilizar esta línea de código, se agrega a un evento onclick y creará una ventana pop-up a la que le asigna el nombre identificador de ventana1. Además, al enlace le ha añadido un target hacia ventana1. De este modo, cuando se clica en el enlace, este carga el documento dentro de la ventana ventana1. Si se fija en esta función, no ha indicado el documento que quiere que abra. Esto ocurre debido a que no es necesario indicárselo, ya que, tan solo está desviando el destino de este enlace a una nueva ventana.

## 2.5. Otros efectos

CSS3 aportan nuevas características con las que puede crear efectos muy profesionales para su web, y todo esto en pocas líneas. A continuación, va a ver un ejemplo del código sobre el que trabaja con CSS:

```

 Deslizar
 Rotar
 Redimensionar

<div id="modal1" class="modalmask">
```

```
<div class="modalbox movedown">

 X

 <h2>Deslizar</h2>

 <p>La ventana modal aparece desde arriba y se desliza hasta su posición.</p>

 <p>Aquí se puede incluir todo tipo de cosas como vídeos, formularios, mapas, etc.</p>

</div>

</div>

<div id="modal2" class="modalmask">

 <div class="modalbox rotate">

 X

 <h2>Rotar</h2>

 <p>Al usar la propiedad transform de CSS3 puede hacer que las ventanas aparezcan rotando.</p>

 <p>No utiliza JavaScript, solo unas líneas de CSS</p>

 </div>

</div>

<div id="modal3" class="modalmask">

 <div class="modalbox resize">

 X

 <h2>Redimensionar</h2>

 <p>Puede redimensionar la ventana hasta que desaparezca.</p>

 <p>CSS3 ofrece múltiples oportunidades con las que crear diversos efectos.</p>

 </div>

</div>
```

En este código la clase modalmask es la que permite aplicar el fondo oscuro, que es frecuente en las ventanas modales, al superponer un div que se extiende sobre el propio contenido. Dentro de dicho div, se pone una ventana a la que aplicar dos clases más. Una sirve para dar estilo a la ventana y otra para crear el efecto. Posteriormente, añade otro link para crear el botón de cierre.

A continuación, vea el CSS que emplee:

```
.modalmask{
Position: fixed;
Font-family: Arial, sans-serif;
Top: 0;
Right: 0;
Bottom: 0;
Left: 0;
Background: rgba(0,0,0,0.8);
z-index: 99999;
opacity: 0;
-webkit-transition: opacity 400ms ease-in;
-moz-transition: opacity 400ms ease-in;
Transition: opacity 400ms ease-in;
Pointer-events: none;
}
.modalmask: target{
Opacity: 1;
Pointer-events: auto;
}
```

En este código el efecto modal está creado por la superposición de un div, el cual es semitransparente, sobre todo el contenido. Para ello, ha añadido la propiedad z-index. Posteriormente, para ocultar el contenido se ha añadido la opacidad 0 y, también, ha desactivado los eventos del puntero del ratón y, así, evita que el ratón reaccione a los elementos ocultos.

A continuación, para que los efectos de animación se inicien y se muestren las distintas ventanas modales, utiliza el selector target que ofrece CSS3. Este selector modifica los estilos del div al que referencia en el enlace. En su ejemplo, lo que hace es restablecer la opacidad y reactivar los eventos del puntero que desactiva anteriormente. Ahora que ya tiene la primera parte de la capa modal, debe dar forma a su ventana. Vea el siguiente ejemplo:

```
.modalbox{
 width: 400px;
 position: relative;
 padding: 5px 20px 13px 20px;
 background: #fff;
 border-radius: 3px;
 -webkit-transition: all 500ms ease-in;
 -moz-transition: all 500ms ease-in;
 transition: all 500ms ease-in;
}
```

Lo que hará a continuación es animar la aparición de estos elementos. Para ello, utiliza el selector target que permite utilizarlo con los elementos hijos también. Vea el ejemplo:

```
MOVEDOWN{
 margin: 0 auto;
}

.rotate{
 margin: 10% auto;
 -webkit-transform: scale(-5, -5);
```

```
Transform: scale (-5, -5);
}

.resize{

Margin: 10% auto;

Width: 0;

Height: 0;
}

.modalmask: target .movedown{

Margin: 10% auto;
}

.modalmask: target .rotate{

Transform: rotate(360deg) scale (1, 1);
-webkit-transform: rotate(360deg) scale(1, 1);
}

.modalmask: target .resize{

Width: 400px;

Height: 200px;
}
```

En este código se establece el estado inicial de cada clase y el estado final mediante el target, por lo que, los estilos se aplicarán al pulsar el enlace en cada ventana modal.

Por último, debe crear un código CSS con el botón que se encargará de cerrar las ventanas.

Vea el ejemplo.

```
.close{

Background: #ef0b0b;
```

```
Color: #ffffff;
Line-height: 25px;
Position: absolute;
Right: 1px;
Text-align: center;
Top: 1px;
Width: 24px;
Text-decoration: none;
Font-weight: bold;
Border-radius: 3px;
}
.close:hover{
Background: #faac58;
Color: #222;
}
```

Este código es muy simple porque si quiere hacer que las ventanas desaparezcan, tan solo, debe cambiar el target a otro enlace, en su ejemplo, cambiará el enlace a #close, el cual no aporta ningún estilo CSS.

En resumen, el código CSS cada vez aporta más características y mucho más avanzadas que antes. Si combina CSS con HTML5, va a tener la posibilidad de crear cosas que antes solo podría haber hecho mediante JavaScript junto con sus distintas librerías. Este aspecto es muy positivo para el desarrollo web, ya que, lo que antes era engorroso de programar, ahora puede hacerlo con unas pocas líneas sencillas en CSS y HTML.

### 2.5.1. Efectos con HTML

Los efectos que puede programar con HTML y CSS son tales como cambiar el color de fondo, el tamaño de fuente, darle determinados efectos al texto, etc. Además, los efectos especiales que

puede crear con HTML producen que el texto aparezca o se mueva de manera especial, de este modo puede captar la atención del usuario en su web. El uso de estos efectos brinda mucho dinamismo a su web y mejora el aspecto de la misma.

Como vio anteriormente, los archivos .js son donde se almacenan los scripts de JavaScript. Estos archivos puede modificarlos cuando y como quiera añadiendo, así, nuevas funcionalidades a su web. El código almacenado en estos archivos es ejecutado por los navegadores.

El código que almacena en estos archivos se constituye por funciones y variables globales que se ejecutarán en la propia página web. De este modo es como hace posible que las funciones sean llamadas desde otras páginas que están por debajo de la principal. Así, ahorra tener que incrustar los scripts dentro del código HTML y, por ende, obtiene un código más limpio y mejor estructurado. Estos archivos son similares a los archivos CSS, la diferencia es que, en vez de trabajar con estilos, trabajan con scripts.

Para utilizar archivos .js externos al código principal va a incluirlo de la siguiente manera en su archivo HTML:

```
<head>

<script src="archivo.js"></script>

</head>
```

La extensión .js es una abreviación de JavaScript, así indica al archivo que el lenguaje de script utilizado en su interior es JavaScript y no otro. Estos archivos se pueden editar en cualquier editor de archivos como, por ejemplo, Visual Code, Atom, Notepad, entre otros.

Anteriormente, cuando utilizaba código JavaScript incrustado en el propio archivo HTML, lo hacía de la siguiente manera:

```
<script>

Function ventana(documento, ancho, alto){

Window.open(documento,"título de ventana","toolbar=0, location=0, directories=0, status=0,
menubar=0, scrollbars=yes, resizable=0, copyhistory=0, width='"+ancho+"', height='"+alto+"');

}

</script>
```

En este ejemplo, utiliza las etiquetas script en HTML para insertar su código. Sin embargo, si lo que quiere es utilizar JavaScript desde un archivo externo, debe hacerlo de la siguiente manera:

```
<html>
```

```
<head>

<script src="script.js"></script>

</head>
```

Además, debe insertar la función requerida en su archivo .js. Vea el ejemplo:

```
Function ventana(documento, ancho, alto){

Window.open(documento,"título de ventana","toolbar=0, location=0, directories=0, status=0,
menubar=0, scrollbars=yes, resizable=0, copyhistory=0, width='"+ancho+"', height='"+alto+"');

}
```

En este ejemplo, puede observar que ya no aparecen las etiquetas `<script>` que aparecían en el archivo HTML.

Al igual que ha hecho esto con código JavaScript, también, puede realizarlo con código CSS, pero eso lo verá en el próximo apartado.

A continuación, va a ver algunos de los ejemplos de archivos .js que puede encontrar en la web. Estos archivos incluyen código JavaScript ya desarrollado que brinda efectos determinados a su web. Vea los ejemplos:

Efecto nieve para su web, tendrá que descargar el archivo snow.js:

```
<mce: script type="text/javascript"

_mce_src="http://.../js/snow.js"></mce:script>
```

Efecto de meteoritos destruyendo la web, debe descargar anteriormente el archivo mynd.js:

```
<mce: script language="javascript">

<!--nd_mode="meteor";nd_dest="massive";nd_control="on";nd_sound="on";-->
</mce:script><mce:script language="javascript" _mce_src="http://.../js/mynd.js"></mce:script>
```

Efecto de fuegos artificiales:

```
<mce:script type="text/javascript"><!--var bits=100; var intensity=7; var speed=20; var colors=new
Array("#03f", "#f03", "#0e0", "#93f", "#0cc", "#f93");
```

Por último, verá el efecto de una estrella volando por su web:

```
<mce:script type="text/javascript" _mce_src="http://.../js/starfield.js" ></mce:script>
```



0

## RECURSO MULTIMEDIA



### 2.5.2. Efectos con CSS

En la web puede encontrar una infinidad de recursos para utilizar los denominados shaders. Para generar los shaders, utiliza lenguajes como Shading OpenGL ES, los shaders están diseñados para programar diversos efectos en su web de manera más sencilla. Los shaders que implementa CSS facilitan programar efectos visuales en su web más fácilmente que sin ellos.

Lo siguiente que va a ver son algunos efectos para el atributo mouse over que utiliza con imágenes. Este efecto se puede utilizar para resaltar imágenes o enlaces. Es un efecto muy fácil de configurar y de añadir a su web. Vea el código HTML y CSS que necesita para implementarlos donde puede configurar los diferentes parámetros según le guste:

#### Efecto UP en imagen

Esquema de cómo quedarían las imágenes posicionadas:

Imagen1	Imagen2	Imagen3	Imagen4
---------	---------	---------	---------

Código HTML:

```
<div class="ej1">

</div>
```

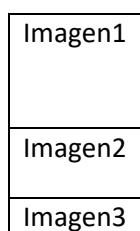
Código CSS para añadir el efecto:

```
Img{
 border: 5px solid #ccc;
 float: left;
 margin: 15px;
 -webkit-transition: margin 0.5s ease-out;
 -moz-transition: margin 0.5s ease-out;
 -ms-transition: margin 0.5s ease-out;
 transition: margin 0.5s ease-out;
}

ej1 img:hover{
 margin-top: 2px;
}
```

### Efecto Zoom en imagen

Esquema de imágenes



Código HTML:

```
<div id="containerEj2">
<div class="ej2">


```

```


</div>

</div>
```

Código CSS:

```
#containerEj2 {

 Width: 300px;

 Margin: 0 auto;

}

#ej2 img{

 Height: 100px;

 Width: 300px;

 Margin: 15px 0;

 -webkit-transition: all 1s ease;

 -moz-transition: all 1s ease;

 -ms-transition: all 1s ease;

 Transition: all 1s ease;

}

#ej2 img:hover{

 Height: 133px;

 Width: 400px;

 Margin-left: -50px;

}
```

### Efecto Giro

Va a programar este efecto para que la imagen gire de manera sutil hacia la izquierda cuando pase el puntero del ratón por encima de ella.

Esquema imágenes:

Imagen1	Imagen2	Imagen3	Imagen4
---------	---------	---------	---------

Código HTML:

```
<div id="containerEj3">

<div class="ej3">

</div>

</div>
```

Código CSS:

```
#containerEj3 {

Width: 800px;

Margin: 0 auto;
}

#ej3 img{

Margin: 20px;

Border: 5px solid #eee;

-webkit-box-shadow: 4px 4px 4px rgba(0,0,0,0.2);
```

```
-moz-box-shadow: 4px 4px 4px rgba(0,0,0,0.2);
box-shadow: 4px 4px 4px rgba(0,0,0,0.2);
-webkit-transition: all 0.5s ease-out;
-moz-transition: all 0.5s ease;
-ms-transition: all 0.5s ease;
}

#ej3 img:hover{
-webkit-transform: rotate (-7deg);
-moz-transform: rotate (-7deg);
-ms-transform: rotate (-7deg);
Transform: rotate (-7deg);
}
```



## RECURSO MULTIMEDIA



### 2.5.3. Efectos con capas

Los efectos con capas tan solo son un elemento rectangular en el que en su interior puede añadir lo que desee como, por ejemplo, párrafos, imágenes, enlaces, etc. Puede añadir uno o más elementos a la vez incluyendo otras capas y tablas de animaciones flash. Todos estos elementos estarán contenidos en un documento HTML. Después de incluir estos elementos en su documento HTML, además, puede modificarlos con código CSS cambiando, de ese modo, las propiedades de los elementos a su antojo. Por ejemplo, el color de fondo, el tipo de letra, el color de letra, las

dimensiones del rectángulo, etc. Vea a continuación cómo se representaría de manera visual una capa:



Dentro de este recuadro es donde añadiría los elementos que precise. Para trabajar con capas, debe conocer sus propiedades las cuales verá en la siguiente lista:

- **Id de capa:** Es el nombre identificador de la capa.
- **Left y top:** Indica la distancia en píxeles que hay entre el borde izquierdo y el superior entre el documento y la propia capa.
- **Width y height:** Indican la altura y la anchura de la capa.
- **Index-z:** Este valor es el número de orden de colocación de las capas. Si una capa tiene un número mayor a otra, se solaparán.
- **Visibility:** Indica la visibilidad de la capa. Puede utilizar cuatro tipos de visibilidad:
  1. Default: Es visible según los criterios del navegador.
  2. Inherit: La capa se muestra si se muestra la página en la que se ubica.
  3. Visible: La capa se muestra, aunque no se muestre la página donde se ubica.
  4. Hidden: La capa está oculta.
- **Imagen de fondo y color:** Indica un color o imagen de fondo de la capa.

Normalmente, se usan las capas para crear menús desplegables donde si pasa el puntero de su ratón sobre el texto del menú, se despliega un menú hacia abajo, hasta ver su contenido totalmente. Cuando quita el puntero del ratón de encima de ese menú, este vuelve a retraerse hasta adquirir su forma original. Vea un ejemplo visual de cómo sucedería esto:

Pulse aquí	Pulse aquí	Pulse aquí
	Contenido 1	Contenido 1
	Contenido 2	Contenido 2
		Contenido 3

Para crear algo parecido a esto debe programar en HTML para brindarle una estructura, en JavaScript para brindar funcionalidad y en CSS para aportarle un estilo atractivo. Vea el ejemplo a continuación:

### Código HTML

```
<div class="contenedor">

 <div class="contenido0">Pulse aquí</div>

 <div class="contenido1">Contenido 1</div>

 <div class="contenido2">Contenido 2</div>

 <div class="contenido3">Contenido 3</div>
```

### Código JavaScript

```
$(".contenido0").hover(

 Function()

 {

 $(".contenido1").slideDown();

 $(".contenido2").slideDown();

 $(".contenido3").slideDown();

 },

 Function()

 {

 $(".contenido1").slideUp();

 $(".contenido2").slideUp();

 $(".contenido3").slideUp();

 });
```

### Código CSS

```
.contenedor{

 Margin: 0 auto;
```

```
Padding: 0;
Width: 300px;
Border: 1px solid black;
Background: White;
}
.contenido0{
Margin: 0 auto;
Text-align: center;
Padding: 0;
Height: auto;
Background: #cccccc;
}
.contenido1{
Margin: 0 auto;
Text-align: center;
Padding: 0;
Display:none
Height: auto;
}
.contenido2{
Margin: 0 auto;
Text-align: center;
Padding: 0;
```

Display:none

Height: auto;

}

.contenido3{

Margin: 0 auto;

Text-align: center;

Padding: 0;

Display:none

Height: auto;

}



### 3. PRUEBAS Y VERIFICACIÓN EN PÁGINAS WEB

En este tema, va a aprender todo lo relativo a las técnicas de verificación y a las pruebas que debe someter a su página para que sea viable para el usuario. También, va a ver alguno de los errores más comunes que puede hallar en los documentos HTML y CSS para que, de este modo, no tenga ningún problema en implementar su web en los diversos navegadores que puede encontrar en Internet.

Además, verá cómo verificar y validar documento HTML y CSS para distintos dispositivos y navegadores.

Todos estos conocimientos en conjunto permitirán desarrollar una página web profesional y totalmente funcional, lo cual será muy positivo para el SEO de su web y para la experiencia de usuario.

#### 3.1. Técnicas de verificación

Como ha visto ya en temas anteriores, sabe que los documentos que conforman una web están dirigidos a los usuarios que van a utilizar esa web. Sin embargo, hay otros muchos elementos intermedios que van a interactuar con esos documentos. Por ejemplo, el hardware y el software de un dispositivo, los navegadores y buscadores. Es por ello, que debe crear su página web pensada para que interactúen usuarios con ella de la manera más fácilmente posible, pero también debe pensar cómo interactuarán los elementos intermedios con la misma. Cuanto mejor optimizadas estén sus webs, mejor se comportarán a la hora de usarlas.

Hay navegadores antiguos que no son capaces de interpretar los estilos CSS, también los hay que no son capaces de interpretar los colores y, además, los que solo son capaces de interpretar texto. Por todo esto, debe ser conscientes que, a la hora de diseñar su web, tiene que abarcar la mayoría de navegadores existentes. Eso proporcionará un mayor número de usuarios que visiten su web. Por lo tanto, su objetivo a la hora de crear una web es hacerlas lo más optimizadas posibles para que los navegadores no tengan ningún inconveniente a la hora de ejecutar la página. Además, también debe tener en cuenta los robots de Internet, los diferentes dispositivos donde los usuarios verán la web, etc.

Si ha fijado hasta ahora, todos los documentos HTML que ha creado a lo largo del curso siempre han tenido una declaración de tipo al inicio del propio documento, la famosa marca `<!DOCTYPE html>`. Esta marca permite que los navegadores interpreten el documento y comprobar si la página cumple con los estándares establecidos. Una vez que hayan extraído la información, adecuarán el funcionamiento para, de ese modo, optimizar la página lo máximo posible. Generalmente, los navegadores que utilizan utilizan dos modos de funcionamiento:

**Standards mode:** Es el modo por excelencia, puesto que, en este modo, el navegador se ajusta de manera estricta a los estándares establecidos actualmente. Por lo tanto, el navegador no reconocería las incorrecciones de la web.

**Quirks mode:** Los navegadores activan este modo cuando detectan que el documento que va a procesar es un documento antiguo. Esto hace que el navegador se haga más flexible ante estos documentos e interprete de manera correcta algunas incorrecciones que eran comunes antes de establecer los estándares actuales.

Las incorrecciones que aceptan los navegadores usando el Quirks mode pueden ser muy diversas, ya que, cada navegador funcionará de una manera determinada. Sin embargo, en el Standard mode sí que tienden a coincidir en la manera en la que funcionan. Esto se debe a que los navegadores funcionan en base a un estándar general. A continuación, vea algunas de las diferencias que hay en los navegadores Opera, Mozilla e Internet Explorer al utilizar el Quirks mode:

Aceptar códigos de colores en hexadecimal sin la necesidad de utilizar la almohadilla "#".

Cuando no indica los valores en los atributos de medidas como height o width, el navegador asumirá que se trata de píxeles.

Interpretar palabras clave y códigos de color de manera correcta, aunque estos estén entrecomillados.

Interpretación distinta de las propiedades que controlan el ancho de los elementos respecto al Standard mode.

Se pueden interpretar los tamaños de letra de manera diferente a lo que se interpreta en el Standard mode.

Como conclusión, debe entender que no es quien va a elegir el modo con el que va a funcionar el navegador, sino que es el navegador quien va utilizar uno u otro modo de funcionamiento respecto al documento que ha creado. Si su documento se asemeja lo máximo posible a los estándares establecidos, el navegador utilizará el Standard mode. En cambio, si su documento no se ajusta a los estándares, el navegador utilizará el Quirks mode.

La conclusión para que el navegador funcione con el modo estándar, se determinará en base a si su documento tiene una declaración de tipo de documento completa. Para ello, se debe incluir la url a la DTD, y que corresponda al estándar de HTML5 o XHTML de manera estricta. En cambio, si la versión de HTML es la 4.01, el navegador podrá elegir el modo Quirks o el estándar dependiendo de si se ha incluido o no la url al DTD. Por el contrario, si la versión de HTML es más antigua, el navegador utilizará siempre el modo Quirks.

Si, por ejemplo, su documento carece de una declaración de tipo de documento, el navegador pasará a utilizar siempre el Quirks mode, ya que, los documentos antiguos se diferencian de los nuevos precisamente en que los antiguos no llevan ninguna marca de tipo de documento.

Por otra parte, si al crear su documento incluye la declaración de tipo de documento, separa el código CSS del HTML y utiliza HTML5 o XHTML de manera estricta, los navegadores siempre

utilizarán el estándar mode. Esto obligará a cumplir con las especificaciones que va a referenciar en la declaración de tipo de documento.



TOME NOTA

***Debe tener en cuenta que la manera en la que el navegador trate sus documentos va a ser influenciado por el modo en que ha entregado esos documentos en el servidor web. Por ejemplo, en Mozilla siempre se utilizará el estándar mode si el tipo de documento es text/html o text/xhtml+xml.***

Si quiere saber en qué modo está funcionando su navegador al ejecutar su documento, lo que debe hacer antes que nada es recuperar ese documento. Para ello, debe introducir la siguiente línea como url:

Javascript:alert (document.compatMode)

Realizar esta acción es muy útil y rápida, ya que, los navegadores actuales ejecutan todos en su mayoría código JavaScript, a no ser que el usuario haya desactivado JavaScript de manera voluntaria. Al ejecutar ese script, el navegador ará una respuesta en la que, si aparece el mensaje CSS1Compat, es que el navegador está funcionando con el estándar mode. Por el contrario, si devuelve el mensaje BackCompat, estará funcionando en Quirks mode.

Para que los navegadores sean capaces de procesar de manera correcta los documentos, además de tener en cuenta el modo en el que deben procesarlos dependiendo de la especificación que ponga al inicio del documento, también debe indicarle al navegador qué codificación de caracteres utilizar para procesar adecuadamente el documento. Como ya sabrá, cada carácter se asocia a un código que el ordenador interpreta posteriormente en código binario. Dependiendo de qué codificación le indique al navegador, asociará esos códigos a los caracteres específicos que alberga su documento.

Por regla general, el servidor web será el responsable de indicar la codificación, por defecto, en la cabecera del documento cuando el mismo es enviado a los usuarios a través de la red. Por lo tanto,

no puede en sí modificar esta acción, ya que, depende de la propia empresa a la que pertenezca el navegador. Sin embargo, tiene dos opciones para indicar la codificación en su documento antes de que sea el navegador quien elija la codificación. Vea ambas posibilidades:

La primera es incluir una marca de identificación XML con el atributo encoding de la siguiente manera:

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
```

La segunda alternativa, que es la que se ha utilizado para el ejemplo anterior, es insertar en la etiqueta head del documento HTML un elemento meta donde indica la codificación deseada de la siguiente manera:

<meta charset="utf-8">

Aun conociendo ambas técnicas, se recomienda utilizar la segunda por temas de compatibilidad con los navegadores actuales. Esto se debe a que la marca de identificación XML causa algunos errores en determinados navegadores como, por ejemplo, en Internet Explorer. El uso de la marca de identificación SML va a impedir visualizar el documento de manera correcta. Por lo tanto, hasta que todos los navegadores no sean capaces de interpretar la marca XML, es preferible utilizar el elemento meta de HTML.

Además, debe recordar que la codificación, por defecto, que usarán los navegadores será utf-8 o utf-16. En ellas, puede diferenciar los subconjuntos ASCII y US. Esto quiere decir que, dependiendo de la codificación, se interpretarán solo los caracteres del inglés o diversos caracteres de las lenguas europeas. En el caso en el que no especifique el tipo de codificación ISO -8859-1, debería escribir los caracteres acentuados en base a entidades. Por ejemplo, cogiendo el ejemplo anterior de la palabra informática, debería escribirla como: Inform&aacute;tica, para que el navegador interprete la letra á de manera correcta.

En el caso de el sistema operativo de Windows, las extensiones de los archivos son una parte esencial para que el sistema reconozca el tipo de archivo que es y lo que alberga en su interior. Sin embargo, en temas de web, la extensión de los archivos no suele ser un indicativo, puesto que el servidor web es el que, en respuesta a la petición de un usuario, añadirá la siguiente línea al documento:

Content-Type: text/html

La cual será procesada por el navegador que, a su vez, será el encargado de procesar la información que hay en el interior de ese documento dependiendo de la indicación de tipo que haya establecido el servidor web. Si el documento a procesar tiene una extensión .html o .htm, el servidor enviará al navegador el tipo text/html. Sin embargo, si el documento tiene otra extensión como, por ejemplo, .xhtml y el servidor web no la reconoce, añadirá la información de tipo de la siguiente manera:

Content-Type: application/octet-stream

Esto causará una confusión en el navegador el cual entenderá que lo que está recibiendo es un programa en vez de un documento. Por lo tanto, el navegador intentará ejecutar el supuesto "programa", intentará guardar el documento o preguntará qué quiere realizar con esa información. Esto sucede de manera similar en otros navegadores o, incluso, en otros dispositivos como, por ejemplo, las PDA y teléfonos móviles.

Resumiendo, el problema por lo que sucede esto es porque los servidores web no están preparados para interpretar la extensión .xhtml y, posteriormente, asociarla al tipo de información especificada en el archivo. Es por eso que a la hora de utilizar archivos con la extensión .xhtml, debe asegurarse de que el servidor web donde lo va a ubicar es capaz de interpretar este tipo. Si por el contrario el servidor es incapaz de interpretar la extensión .xhtml, la única alternativa que tendrá es cambiar la configuración del propio servidor o utilizar las extensiones .html o .htm. Después de llevar a cabo

esta solución, los navegadores serán capaces de interpretar documentos donde se haya usado código xhtml.

Tanto los documentos HTML, como los documentos XHTML deben iniciarse con la declaración DOCTYPE para indicarle, de ese modo, al navegador el lenguaje en el que está escrito el documento. Este punto es muy importante porque además de cumplir con los estándares del W3C, los navegadores contemporáneos son capaces de utilizar distintos modos de renderizado dependiendo del DOCTYPE utilizado. Por el contrario, si no utiliza el DOCTYPE va a repercutir de manera negativa al modo de renderizado de la página en el navegador. Vea un ejemplo de cómo comenzar un archivo HTML:

```
<!DOCTYPE HTML PUBLIC "-//w3c//dtd html 5.1 Transitional//en"
"http://www.w3.org/tr/html5/loose.dtd">
```

Para realizar pruebas de verificación de manera sencilla y rápida, puede encontrar múltiples herramientas gratuitas en Internet. En este apartado va a aprender cómo realizar este tipo de revisiones a sus webs para ver si cumplen los términos de referencia y los estándares mínimos que están referidos en la normativa vigente actual. Para ello, va a ver las múltiples opciones que existen para verificar que es el propietario de un sitio web. Vea las siguientes opciones:

La primera opción es añadir una metaetiqueta en la página principal de su sitio web. Esto demostrará que tiene acceso a los archivos originales del sitio. Para llevar a cabo esta acción, debe ser capaces de editar el código HTML de las páginas creadas en su sitio web.

La segunda opción es que suba un archivo HTML con el nombre con el que haya denominado a su servidor. Si quiere usar este método de verificación debe poder subir archivos a su servidor.

La tercera opción es poder verificar su sitio web mediante el proveedor DNS. Si quiere verificar su sitio mediante este método, debe registrar el nombre de su dominio en el registro de DNS.

La cuarta, y última opción, es utilizar el código de Google Analytics que utiliza para comprobar el seguimiento de su sitio web. Si desea utilizar esta opción, debe ser el administrador de su cuenta de Google Analytics para poder utilizar el código de seguimiento que da Google Analytics en su página.

Debe saber que la herramienta de Google para webmasters comprueba el archivo, el código, la etiqueta o el registro que especifica en su sitio, estén presentes. Si todos estos requisitos están presentes, estará considerado el propietario del sitio y mostrarán información detallada sobre el mismo. Si en vez de hacer lo anterior, acceda a estas herramientas directamente a través de un proveedor de alojamiento web, ya habrán verificado como propietarios del sitio y puede acceder a toda su información sin hacer ningún tipo de verificación extra.

Por otro lado, también puede crear un sitio web con la herramienta Google Sites, lo cual verificará, también, de forma automática. Lo único que debe tener en cuenta es que debe crear el sitio web con la misma cuenta con la que vaya a utilizar las herramientas de webmaster de Google. Si sucede que

no aparece su sitio en las herramientas para webmasters, debe añadirlo con la opción de Añadir un sitio. De este modo, el sitio se añadirá y se verificará automáticamente su propiedad. Debe tener en cuenta que, si tiene sitios creados anteriormente, no se verificarán de manera automática. Para ello, debe realizar el método de añadir la metaetiqueta que ya se explicó anteriormente.

En el caso en que quiera verificar la legitimidad de una web, debe comprobar obligatoriamente su certificado digital. El certificado digital es un elemento mediante el cual un tercero garantiza que la página es de la entidad que realmente dice ser.

En los navegadores con versiones actuales, se ha implementado la interpretación de certificados mediante un código de colores, es decir, son tan solo mirar el color de la barra de navegación puede comprobar la legitimidad del sitio. Esto hace que todo el proceso de verificación de legitimidad sea más sencillo e intuitivo. Dependiendo del color que se utilice en la barra, puede distinguir dos niveles de confianza. Véalo a continuación:

**La página es totalmente de confianza.** En el caso en el que la barra de direcciones sea de color verde, puede estar totalmente seguros de que la página es de la entidad quien dice ser.

La página puede ser de confianza o no. En el caso en el que la página empiece por el protocolo https, pero la barra de direcciones no esté en verde, debe de considerar algunas cosas para saber si está ante una página legítima o no. Por lo tanto, para poder utilizar la página con la certeza de que es segura debe de asegurarse de que la dirección de la web que va a visitar pertenece realmente a la entidad que dice ser. Además, debe de fijarse minuciosamente que la dirección de la web a visitar esté escrita de forma correcta. Esto se debe a que muchos crackers suplantan la identidad de las páginas utilizando direcciones parecidas a la original para robar datos, dinero, contraseñas, etc.

Si revisa todos estos datos y está seguro de que la página es segura, puede visitarla sin ningún problema. A continuación, vea cómo ver este tipo de páginas en los diferentes navegadores. En Internet Explorer, por ejemplo, aparecerá un candado con fondo azul. Si pulsa este candado, aparecerá un certificado que garantiza una conexión segura y una legitimidad real. Por otro lado, en Mozilla Firefox, verá que el ícono de la página aparece con el nombre de la entidad y con fondo azul, lo mismo que ocurría anteriormente con el fondo verde. Por último, en el navegador Safari puede ver que aparece un candado en la barra de direcciones, pero no aparece ni el nombre de la entidad, ni el fondo verde.

Cuando crea una página web, debe siempre que someterla a pruebas de adecuación a resoluciones de pantallas distintas, ya que, actualmente existen diversos dispositivos con los que los usuarios pueden acceder a su web y, estos dispositivos, tienen tamaños y resoluciones de pantalla muy diferentes. Por ejemplo, no tiene el mismo tamaño la pantalla de un móvil que la de un ordenador. Sin embargo, tanto los usuarios de móviles como los de ordenador deben de ver su web de manera parecida, si no podría perder visitas.

Aparte de comprobar las diferentes resoluciones de la web que solo afecta a las dimensiones de la misma, también debe comprobar la resolución del color. Esto se debe a que su sistema utilizará un formato de 24 a 32 bits de color real. Sin embargo, los usuarios podrían tener otras configuraciones

---

de color, que pueden abordar miles de colores. Entonces, para adaptar los documentos a esas configuraciones, debe adaptar su resolución mediante la configuración del sistema y, de este modo, puede comprobar los resultados. De modo que, cuantas más pruebas haga en diferentes configuraciones, equipos y sistemas operativos, mejor resultados obtendrá al aplicar el color a su web, ya que, habrá obtenido más referencias de cómo quedarán los colores en diferentes contextos.

Además, debe ser conscientes de que uno de los problemas con los que más veces se topa es que si ha creado una página web no responsive, va a notar que, al cambiar la resolución de la pantalla, todos los elementos que hay en ella se van a desbaratar por completo. Esto sucede por haber creado la página con una distribución de tamaño fija. Al haber dado medidas en píxeles, los elementos se distribuirán de manera fija por la pantalla, por lo que, si la pantalla es más pequeña, faltarán elementos que visualizar, por lo que el usuario deberá desplazarse por la página con las barras de desplazamiento, lo cual es algo bastante engorroso. Y, en el caso de que sea más grande, verá mucha distancia entre los bordes de la pantalla y los elementos.

Por lo tanto, para realizar una página web responsive capaz de adaptarse a cualquier resolución, debe establecer los tamaños y distancias entre los elementos, utilizando porcentajes para el reparto de espacio en la página y, para el tamaño de letra, usa el elemento em. Hoy en día con tal cantidad de dispositivos con pantallas de tamaños totalmente diferentes, debe, siempre que sea posible, crear webs de tipo responsive.

La mayoría de los navegadores que utilizan los usuarios son capaces de ejecutar scripts y hojas de estilo, incluso aquellos navegadores son capaces de ello. Sin embargo, hay algunos que no son capaces de realizar esta acción y, por lo tanto, no tendrán la capacidad de interpretar el código CSS, por lo que, los usuarios podrán ver el contenido de su web, pero no sus estilos. Por otro lado, existen programas que tampoco son capaces de ejecutar los scripts creados con JavaScript que haya incluido en su web, ni incluso de visualizar imágenes. Esto se puede deber a que solo son capaces de funcionar en modo texto. Además, también puede encontrarse con el caso en que un navegador no sea capaz de interpretar una característica muy específica como, por ejemplo, el uso de :hover en elementos que funcionan como hipervínculos. Esta característica es muy importante, ya que, permite crear un botón tan solo cambiando el borde de la sección de un div mediante la siguiente línea de código:

Div:hover {propiedades}

Al utilizar esta línea de código, por ejemplo, tendrá problemas al utilizar el navegador Internet Explorer, ya que, este navegador solo permite utilizar :hover con hipervínculos y no con elementos. Por lo tanto, cuando se encuentra en estos casos, debe hacer que su documento sea capaz de degradar su diseño o comportamiento de la manera más correcta posible, haciendo que no pierda nunca la información que quiere que transmita el mismo. La manera más sencilla de realizar esta acción es probar el documento en diversos navegadores.

En definitiva, si una web que contenga la línea de código, anteriormente mencionada, se ejecuta en el navegador Internet Explorer, puede decir que el funcionamiento de la página en sí se ha degradado, puesto que, la misma página en otros navegadores va a tener un comportamiento más

---

dinámico que en IE. Sin embargo, esa degradación no va a afectar de manera significativa al usuario, ya que, la información que le llega es la misma.

Como decía anteriormente, no todos los navegadores son capaces de interpretar adecuadamente el lenguaje CSS, por lo que, en estos casos la página perdería toda la información que contenga acerca de los estilos. Para asegurarse de que solo se degradará el aspecto de la página, sin afectar gravemente a la distribución de los elementos en la misma, debe haber estructurado el contenido de manera lógica. Esto quiere decir, que, aunque utilice el código CSS para posicionar un elemento en un punto determinado, debe también programar en HTML la distribución de los elementos siendo conscientes de cómo se vería si los elementos se mostraran de manera secuencial. Para comprobar cómo quedaría la página, que está creando, si el navegador fuera incapaz de interpretar las hojas de estilo, sencillamente debe eliminar los elementos link que referencian a la hoja de estilo. Esto hará que al ejecutarse su página no cargue los estilos y, de este modo, puede comprobar cómo se visualizaría sin los mismos.

De esta manera, los dispositivos con poca potencia de proceso y con poca memoria, podrán beneficiarse de que sus páginas puedan mostrarse sin la necesidad de cargar hojas de estilo, ya que, esto conllevará a tener que esperar menos tiempo de carga. Estos dispositivos pueden ser móviles de baja gama, ordenadores antiguos, tablets de gama baja, PDAs, etc. Estos dispositivos lo que harán es en vez de recuperar la hoja de estilo, aplicarán unos atributos determinados a los diferentes elementos como, por ejemplo, tipos de encabezados diferentes, enlaces, párrafos, etc.

Sin embargo, en algunos casos se encuentra con que los navegadores no son capaces de interpretar el código CSS, pero, además, tampoco son capaces de mostrar imágenes, y algunos de ellos no tienen ni siquiera la capacidad de mostrar colores. Es por ello, que existen una gran variedad de configuraciones y, sobre todo, en los sistemas Unix donde muchos usuarios acceden a Internet a través de terminales que solo pueden mostrar texto. Por lo tanto, si la página está bien diseñada todos estos usuarios no tendrán ningún problema para visualizarla. Para comprobar todas estas variantes, se recomienda tener Lynx instalado en su equipo y probar sus documentos con esta aplicación antes de subirlos al servidor.

### **3.1.1. Fundamentales**

Aunque la creación de documentos JavaScript, CSS y HTML puede realizarse mediante el uso de editores de texto convencionales, si quiere dedicarse al desarrollo de páginas web, debe utilizar herramientas más profesionales y sofisticadas para ello. La razón por la que debe utilizar herramientas más profesionales es porque este tipo de herramientas pueden ayudarnos con la escritura del código de manera que diga en tiempo real qué errores ha cometido en cuanto a la sintaxis. También, puede ayudar asistiendo en la propia escritura, añadiendo trozos de código, puede asistir con una lista de atributos disponible para un elemento determinado, etc. Delegar este trabajo a la herramienta, permite centrarse solo en el contenido que va a albergar la página y su formato.

Además de utilizar o no herramientas específicas de desarrollo web, algo que debe comprobar siempre en los documentos que cree es si cumplen o no con los estándares establecidos. Es por ello, que debe utilizar estas herramientas específicas para validar los documentos que desarrolle. Tiene

que ser conscientes de que, si los documentos que desarrolla cumplen estrictamente los estándares requeridos, tendrá muchos menos problemas con los diversos navegadores a la hora de visualizar sus documentos. Y, de este modo, los usuarios que utilicen tecnologías más o menos potentes, quedarán satisfechos cuando naveguen por su web.

Por otra parte, la mayoría de herramientas de validación de documentos HTML, CSS y JavaScript se encuentran de manera online, es decir, la manera de validar sus documentos de la manera más sencilla posible es coger la dirección url de donde se alojan sus documentos y copiarlas en la página donde se encuentra la herramienta de validación de código. Además, también hay herramientas de validación que son programas ejecutables en su ordenador como, por ejemplo, la aplicación Tidy que es capaz de solucionar los errores más comunes que pudiéra tener en su código. Esto hace que ahorre tiempo y trabajo a la hora de desarrollar una página web.

Finalmente, si sus documentos son validados correctamente, el W3C ofrece unos logotipos que puede añadir en su página para señalar, de este modo, a los usuarios que su página cumple con los estándares requeridos. Añadir estos logotipos a su página hace que los usuarios puedan ver la calidad de la misma y, por lo tanto, sea más profesional y confiable para los mismos.

Para hacer estas comprobaciones en sus páginas puede optar por dos opciones:

### **De manera manual**

Esta manera de realizar la comprobación, como su nombre dice, es de manera totalmente manual y se realiza navegando a través de las páginas del propio sitio web. Antes que nada, debe construir un índice para no saltarse ningún contenido. Una vez haya construido ese índice, debe verificar cada uno de los elementos que contiene la página. Para ello, analice cada recoveco de la página de forma exhaustiva. A continuación, va a ver una lista donde verá los elementos que debe analizar:

- Verificar la lista de chequeo de accesibilidad.
- Verificar la ortografía y redacción de los textos de la web.
- Verificar los hipervínculos y enlaces.
- Verificar la existencia real de los archivos que se hallen adjuntos.
- Verificar las imágenes que alberga la página.

### **De manera automática**

#### **Verificar plataforma y Cross-Browser:**

Cuando habla de Cross Browser está refiriéndose a la capacidad que tiene una página de poder ser mostrada en cualquier tipo de navegador.

Todos los usuarios prefieren un navegador determinado, aunque sepa que hay otros navegadores mejores que el que utiliza diariamente. Es por ello que, es importante que su web pase la verificación de los navegadores más utilizados y que éstos puedan mostrarla sin ningún problema permitiendo a los usuarios navegar a través de la página fluidamente. Desafortunadamente, este es uno de los

problemas más comunes que ocurren a la hora de desarrollar una web, que el diseño no sea lo suficientemente bueno que la web sea incapaz de mostrarse en cualquier navegador. De nada sirve crear un diseño visualmente extraordinario, cuando la mayoría de usuarios no pueden visualizarlo. Es por ello que una de las pruebas más importantes que debe pasar su página es que verifique que su web es capaz de funcionar en los múltiples navegadores disponibles. Y, sobre todo si estos navegadores son conocidos.



***No tener instalados todos los navegadores en su ordenador no excluye de que deba probar su sitio web en todos ellos.***

**TOME NOTA**

#### **Verificar que funcionen los enlaces:**

Es imperativo verificar que todos y cada uno de los enlaces de su web son funcionales y que dirijan hacia el sitio donde deben dirigirse. Por lo tanto, aunque su web esté repleta de un gran número de enlaces, debe pulsarlos uno por uno y asegurarse de que funcionan correctamente, ya que, si no es así, sus usuarios podrían llevarse una mala experiencia con su web. Imagine que tiene una tienda online de ropa que cuando el usuario pulsa en el menú para ver, por ejemplo, zapatos, este enlace no lo redirija a los zapatos sino a las camisetas. En esta ocasión podría perder un cliente potencial. Es por ello que, antes de lanzar su sitio web oficialmente a Internet, debe obligatoriamente que realizar esta prueba en cada enlace y en cada elemento que enlace hacia otro, por ejemplo, vídeos o fotos. Además, también tendrá que verificar que todos los elementos de su web funcionan de manera correcta y que, por lo tanto, su web está lista para que los usuarios naveguen a través de ella, puesto que, los buscadores califican este tipo de cosas para posicionar la web.

#### **Contabilizar el tráfico de usuarios mediante estadísticas:**

Debe asegurarse antes del lanzar una página web a Internet, que tiene algún programa de análisis de tráfico de usuarios funcionando en su web. Estos programas lo que hacen es dar la idea de qué buscan más sus usuarios y cómo interactúan con los elementos de la propia web. Si sabe utilizar este tipo de herramientas de manera correcta, puede hacer que el tráfico de su web cada vez sea más grande. Esto ayudará a posicionar la página en los buscadores y ayudará a conocer en qué están interesados sus usuarios. Una herramienta muy conocida para realizar esta tarea es Google Analytics.

#### **Obtener un código de validación:**

Es importante que su web tenga un código válido, ya que, tenerlo o no tenerlo puede hacer la diferencia entre que su web esté mejor o peor posicionada. Los códigos válidos, también, aportan profesionalidad a la web y la confianza de los usuarios hacia ello. Además, ayuda con el SEO, ya que,

los buscadores van a posicionar de manera primordial las webs que sean más confiables y profesionales.

Para poner un ejemplo, imagine que tiene un cubo defectuoso de fábrica, entonces cuando echa agua en él, se escapa por las grietas del cubo. Tener una página web sin un código de validación es tener una página web defectuosa e incapaz de funcionar correctamente. Al obtener un código de validación, se asegura de que su página no tendrá problemas en un futuro.

#### **Posibilidad de obtener versión para imprimir:**

En el caso de que su web sea de artículos, tutoriales de cursos, ejercicios, o cualquier contenido que esté basado en texto e imágenes, debe haber creado para ello una hoja de estilos. Debe tener en cuenta que, para el usuario es una mala experiencia si intenta imprimir la versión impresa de su web y que, por error en el diseño de la misma haya pasado por alto verificar que se pueda imprimir todo el contenido y, el usuario tan solo logre imprimir el encabezado y algunos fragmentos de texto, en vez de la página completa. Es por ello, que debe crear una hoja de estilo y probarla en diversos navegadores y en múltiples circunstancias para asegurarse de que el diseño de la web no aparezca de manera errónea en alguno de los escenarios y, de ese modo, evitar futuros problemas.

En definitiva, lanzar su web a Internet debe de ser un proceso sencillo siempre y cuando, tome el tiempo de verificar bien todo el contenido y su funcionamiento. Piense que, aunque invertía más tiempo para hacerle pruebas a su web, será tiempo invertido en un futuro y usuarios satisfechos con su página.

#### **3.1.2. Técnicas HTML**

Los documentos HTML, además de contener información para mostrar a los usuarios y estilo para que se vea más atractivo, también necesitan incluir información sobre sí mismos. A esta información, la denomina meta información o, en inglés, metadata. Este tipo de información se pone en elementos específicos como, por ejemplo, el elemento title del HTML. Cada navegador utiliza este elemento de una manera determinada. Otro de los elementos que va a utilizar para incluir meta información al documento es la etiqueta meta. Esta etiqueta ha aprendido a utilizarla tan solo para añadir el tipo de codificación del documento. Es decir, en su caso utf-8. Vea a continuación cómo utilizar esta etiqueta en su totalidad:

```
<meta name="propiedad" content="valor"/>
```

```
<meta http-equiv="propiedad" content="valor"/>
```

Si quiere validar sus documentos HTML y XHTML puede echar mano de la herramienta que proporciona el [W3C](#). Esta herramienta puede identificar el tipo de la página mediante la declaración de tipo de documento. De esta manera, comprueba que el código es coherente con la DTD que se indica en esa declaración y, así, comunica los fallos que pueda tener el documento.

Para utilizar esta herramienta, debe conocer sus partes y ver qué opción se ajustan más. Para ello, va a ver las dos partes que constituyen esta herramienta:

- **Validate by file upload:** Utiliza esta opción si su documento web no se encuentra alojado en un servidor con acceso a Internet. Esta situación es la más común, por lo que, tan solo debe enviar su archivo al servidor de validación de la propia herramienta. Para ello, pulse el botón Examinar, elegir el archivo y, posteriormente, pulse el botón Check.
- **Validate by url:** Esta opción es la que utiliza si el archivo de su web está alojado en un servidor accesible a Internet. Para utilizar esta opción, tan solo tendrá que introducir la url completa de su web y pulsar el botón check.

Para utilizar esta herramienta y que sea de utilidad, sus documentos deberán albergar, de manera obligatoria, la declaración de tipo de documento y el tipo de validación de caracteres que usa el propio documento. Esto sucede de este modo porque todos los documentos que no contengan estos dos elementos, sencillamente, no pasarán la validación. Esto se debe a que cuando suba el documento a la herramienta no existe un servidor que entregue el documento y que, por lo tanto, indique el tipo de codificación de caracteres por defecto. Por lo tanto, el resultado será que el servidor asuma que el documento tiene un tipo de codificación determinada que no concuerde con los caracteres que realmente utiliza el documento y, por ende, el servidor no pueda interpretar estos caracteres.

Por otra parte, si pulsa en la opción Extended File Upload Interface, accede a una página en la que puede encontrar diversas opciones que van a permitir controlar de mayor manera el proceso de verificación del documento. En esta sección encuentra elemento Encoding, el cual va a permitir elegir la codificación que interese entre una lista bastante amplia de tipos de codificación. Además, también puede elegir, si así lo necesita, un elemento de la lista Doctype para poder indicar el tipo de documento a validar. Elegir un elemento de la lista Doctype solo es necesario si su documento no lleva la declaración de tipo de documento. También, podría incluir el elemento, aunque su documento lleve la declaración de tipo, si quiere comprobar la validez del mismo respecto a una especificación diferente. Por ejemplo, puede comprobar si un documento HTML5 estricto, se ajusta a la DTD de un XHTML 1.0.

En esta herramienta, además de las opciones principales que ha visto, existen una serie de opciones que permiten configurar la validación de la forma que más convenga. Vea a continuación la lista de opciones disponibles que tiene para configurar:

- **Verbose Output.** Esta opción, normalmente, se encuentra marcada por defecto. Esta opción se utiliza para que la herramienta facilite una descripción, lo más detallada posible, sobre los errores que contenga su documento.
- **Show Source.** Esta opción hace que se adjunte el código fuente al informe de validación que se genera como resultado de la ejecución de la prueba de validación.
- **Use Fallback instead of Override.** Esta opción está asociada con las listas Encoding y Doctype. Si marca esta opción está indicándole a la herramienta de validación que utilice las opciones de codificación y tipo de documentos que ha seleccionado previamente en las listas solo en caso de que en el documento originalmente no se haya elegido otro tipo.

- **Show Outline.** Esta opción muestra un índice del documento que se genera en base a los contenidos que alberga en los diferentes niveles del encabezado. El índice se añadirá al informe de validación.
- **Exclude Attributtes.** Esta opción tiene como finalidad excluir la información, relativa a los atributos, del árbol del documento. Esta opción se utiliza de manera conjunta con la opción que va a ver a continuación.
- **Show Parse Tree.** Puesto que, para examinar la estructura del documento, la herramienta de validación genera un árbol que contiene todos sus elementos, si marca esta opción va a hacer que el árbol aparezca en el informe resultante de la validación.
- **Validate error pages.** Al elegir esta opción, está permitiendo que todas las páginas puedan ser validadas, ya que, si no elige esta opción y su web lanza un error, la herramienta de validación los mostrará la página de error, pero no validará nada.

De todas estas opciones, puede marcar tantas como necesite. Tan solo debe marcarlas y, a continuación, pulsar el botón de validación y examinar el resultado.

El informe de validación que proporcionará la herramienta de W3C va a indicar el nombre del archivo, la codificación, el tipo de documento y el contenido identificado en él. Si el documento pasa la validación y resulta ser válido, aparecerá el título con un fondo azul donde indicará el lenguaje de programación y su versión. Además, aparecerá un enlace sobre la especificación correspondiente del lenguaje.

Continuando con el documento, aparecerá el logotipo de W3C como un elemento de imagen, la versión de XHTML o HTML a la que se ajusta su documento y una marca de validación. Además, se ofrece rá un código que puede incluir en su página web para señalar que su web cumple con los estándares. También, ese código que ofrece n actúa de hipervínculo de manera que, cuando lo pulsa, volverá a validar la página. Esto es muy útil cuando hace algún cambio en la misma y necesita volver a validarla.

Por otra parte, si ha marcado en la validación las opciones Show Outline, Show Parse Tree o Show Source, el informe mostrará el código fuente de la página, el árbol de elementos o el índice completo del documento.

Aparte de los elementos que ha visto anteriormente, si el documento no supera la validación, se mostrará en el informe de validación el fondo en rojo en vez de en azul, indicando que no se trata de un documento válido. Además, aparecerá debajo una lista con cada uno de los errores que contiene el archivo, indicando en cada error la línea y columna del archivo donde se ha encontrado dicho error. También, se mostrará una descripción corta del error. En el caso en que haya dejada marcada la opción Verbose Output, cada uno de los errores vendrá con una descripción que indicará cuál es el error y las posibles soluciones para resolverlo.

Por otro lado, tiene la herramienta Mobile Checker, que está diseñada para validar sus documentos web de cara a los dispositivos móviles. Esta herramienta permite identificar los errores y deficiencias en su diseño web que pueden causar problemas a la hora de visualizar su web y, por ello, dificultar el acceso a la misma en dispositivos móviles.

## Verificación de interfaces

Esta prueba de verificación de interfaces permite probar los aspectos gráficos de su sitio web. De este modo, puede saber si el despliegue de sus páginas es correcto. Vea a continuación una lista de los elementos que deben ser verificados de manera obligatoria:

### **Plug-ins:**

Si en su web necesita ciertos elementos interactivos que requieran algún tipo de software extra para funcionar, debe de añadir siempre un enlace para aquellos usuarios que no tengan instalados estos *plug-ins*, puedan hacerlo.

Por ejemplo, si su página utiliza elementos Flash, actualmente, los navegadores permiten descargar este *plug-in* de manera automática siempre y cuando la página tenga la codificación adecuada. Es por ello que, debe siempre hacer pruebas desde un navegador que no tenga ese *plug-in* instalado y, de ese modo, comprobar que esto ocurre así realmente.

### **Consistencia de la interfaz:**

Para que la interfaz de una página web sea consistente, debe ofrecer a los usuarios una buena experiencia de usuario. Por ejemplo, los menús de su web siempre deberán aparecer en el mismo lugar, es decir, no puede poner en la página principal el menú arriba, y cuando el usuario pulse una opción que aparezca en la siguiente página el menú abajo. Por otro lado, los listados deben tener un diseño acorde unos con otros y, ese diseño, mantenerse en todas las páginas del sitio. Y, por último, los colores del sitio deben mantenerse iguales y con el mismo diseño. No es correcto poner unos colores en la página principal y en las demás otros colores diferentes.

### **Ancho de la interfaz:**

En el caso en que haya diseñado la interfaz con un ancho determinado, debe comprobar esa interfaz con diferentes tipos de anchos para, de ese modo, ver cómo afecta las diferentes dimensiones a la propia interfaz. De esta manera, será conscientes de cómo afectará esto a la experiencia de usuario. Además, debe utilizar una herramienta llamada Lynx para obtener resultados de cómo se comportaría su interfaz si los usuarios acceden a ella a través de un navegador que solo es capaz de interpretar texto.

Se recomienda tener un archivo log en el servidor para que muestre la manera en la que los usuarios acceden a su web. De esta manera, puede saber qué configuración de pantalla está teniendo problemas y cuál no. Y, de esta forma ser capaces de resolverlos para que ningún usuario tenga una mala experiencia en su web.

### **Interfaces y navegadores:**

Los diferentes navegadores web no interpretan de la misma manera el código con el que están codificados los documentos web, aunque estos sigan los estándares estrictamente. Debido a esta

razón, debe hacer pruebas con diferentes navegadores y ver cómo se visualizan los contenidos en todos ellos.

Estas pruebas, mínimo, deberían realizarse en los navegadores más conocidos como Internet Explorer, Google Chrome, Opera y Mozilla Firefox. Si realiza las pruebas en los navegadores anteriormente nombrados, habrá cubierto un gran espectro. En esta prueba, debe revisar cómo se despliegan los diferentes elementos que componen su página y, de ese modo, asegurarse de que las posiciones de los elementos en cada tipo de navegador cumplen con el diseño original.

#### **Interfaces y Sistemas Operativos:**

Como ya vio anteriormente, los diferentes Sistemas Operativos pueden mostrar las páginas web de manera diferente. Por esta razón, debe saber qué Sistemas Operativos son los más usados entre los usuarios a los que pretende llegar y hacer pruebas de despliegue de elementos en estos Sistemas Operativos.

Debe saber que los usuarios pueden visualizar su contenido además de en el Sistema Operativo de Microsoft, en Macintosh, Linux o Android. No debe olvidar que existe un gran espectro de Sistemas Operativos y estos que ha nombrado anteriormente son solo los más conocidos.

#### **Imágenes a escala:**

Debe asegurarse de que las imágenes que utilice en su sitio web no se muestren en tamaño reducido de manera artificial, es decir, añadir imágenes con un gran tamaño y que, mediante código, se modifiquen sus dimensiones. Haciendo esto obtendrá una web lenta y pesada, por lo que, la experiencia de usuario no será positiva.

Para realizar comprobaciones de si esto está sucediendo, debe acceder a las propiedades de las imágenes. Para ello, hará clic derecho sobre la imagen y en el menú que se nos abre, pulsar Propiedades. En los datos de las propiedades nos aparecerá lo que pesa la imagen. Si está utilizando la imagen para un ícono o algo pequeño no debería sobrepasar los 5Kb. En cambio, en el caso de que esté utilizando la imagen como imagen mediana, no se deberían sobrepasar los 25 Kb.

Además, también debe considerar el peso de la propia página con todos sus elementos incluidos. El peso total de la página no debería sobrepasar los 100 Kb.

#### **Imágenes sin ALT:**

Para crear una página que cumpla con las normas de accesibilidad, necesita que todas las imágenes que se muestren en la misma incluyan el atributo ALT con un texto alternativo que haga una pequeña descripción de la imagen.

Para comprobar si las imágenes de una web cumplen con este requisito, debe posicionar el puntero del ratón sobre una imagen y se deberá desplegar un recuadro amarillo con un pequeño texto. Si,

por el contrario, esto no ocurriera, significaría que la imagen carece de este atributo y, por lo tanto, debería corregirla para que cumpla los requisitos.

En el caso de que esté acostumbrado a utilizar herramientas para webmasters, seguramente, sepa cómo verificar sitios webs. Para ello, tan solo necesite añadir una metaetiqueta o un archivo HTML al sitio web a verificar. Seguidamente, pulsa el botón y ya habrá verificado su sitio web.

Sin embargo, existen otras maneras de verificar las webs de manera más sencilla y fiable. Lo primero es que se ha mejorado el método de verificación con metaetiquetas. Anteriormente, la verificación con metaetiquetas se basaba en la dirección de su cuenta de Google. Esto significaba que, si cambia de cuenta, las metaetiquetas también cambiarían. Esto quiere decir que aparecerían sitios web no verificados, ya que, si había cambiado de dirección de correo electrónico y no había cambiado las metaetiquetas de ese sitio, ya no aparecerían como verificado. Es por ello que, se ha creado una nueva versión para realizar la verificación con metaetiquetas, la cual, ahora, no se relaciona con la cuenta de correo electrónico. Gracias a este cambio, si cambia de cuenta de correo el sitio web no dejará de estar verificado.

Además, también se ha mejorado la verificación con archivos HTML. Por ejemplo, anteriormente, si su sitio web devolvía, para una url que no existía, un código de estado distinto a 404, no podría haber usado este tipo de método de verificación para su web. Ya que, un sitio web con la configuración correcta debe devolver en este caso un código 404. Sin embargo, muchos sitios web tienen problemas para cumplir este requisito. Por ello, se ha hecho un proceso más sencillo para realizar esta verificación. Para ello, se han eliminado las comprobaciones de url no existentes. Con el nuevo proceso, simplemente, tendrá que descargar el archivo HTML que proporcionan y, a continuación, subirlo al sitio web que quiere verificar, sin modificar para nada el archivo. A continuación, se comprobarán los contenidos del archivo, y si son correctos, habrá finalizado el proceso de verificación del sitio.

En el caso de que su sitio haya sido verificado con el método antiguo, no debe preocuparse porque las verificaciones se mantendrán intactas y, por lo tanto, seguirán siendo válidas.

Por otra parte, existen sitios web que ayudan a verificar sus webs. Esto se debe a que esos sitios webs ponen las metaetiquetas o el archivo HTML de manera automática. Aunque, seguramente estos sitios deban actualizarse para que puedan funcionar con los nuevos métodos que ha mencionado anteriormente. Como ejemplo, vea el sitio web de Google Sites que aún no está actualizado y, por lo tanto, es incapaz de utilizar los nuevos métodos de verificación con metaetiquetas.

Se prevé que en un futuro las mejoras para la verificación de sitios web en las que se están trabajando actualmente, sean capaces de mostrar, al resto de propietarios, las direcciones de correo electrónico de todos los propietarios verificados para un sitio específico. Esto se debe a que, gracias a esta actualización, será más fácil la gestión de los sitios web donde haya más de un propietario verificado. Por lo tanto, si quiere que su dirección de correo actual no se muestre, lo suyo es crear una dirección nueva y modificarla en su sitio web para cuando se incluya esta actualización no nos afecte.

---

## **Subir un archivo HTML**

Para verificar una propiedad de dominio y activar, así, Google Apps para su dominio, puede subir un archivo HTML al servidor web donde se encuentra su dominio. Para ello, como es lógico, debe tener acceso al servidor y sus archivos.

En el caso de que su dominio no disponga de servidor web, por ejemplo, si nunca ha configurado un servidor para su dominio, o si, por ejemplo, utiliza un sistema de gestión de contenidos que no da la opción de subir archivos al servidor, tendría que verificar su propiedad de dominio otra forma.

Siguiendo con la explicación de cómo subir un archivo HTML a su web, el archivo que suba deberá estar disponible en su dominio simple y desde Internet. Es decir, que su dominio debe ser visible sin el www en la url.

Esto significa que, su dominio podría verse desde <http://dominio.com/nombrearchivo.html>, y no desde <http://www.dominio.com/nombrearchivo.html>. Debe recordar que el archivo HTML debe de estar disponible públicamente y no debe estar limitado solo a su red local.

## **Generar y subir archivo HTML**

Para generar un archivo HTML de verificación, tendrá que registrarse en Google Apps y seguir las instrucciones para verificar la propiedad de dominio. Para ello, elegir la opción de verificación cargar archivo HTML. En el caso de que este método no salga como recomendado, debe ir a la pestaña de Métodos alternativos.

Para descargar el archivo de verificación HTML, debe pulsar la opción de este archivo de verificación HTML. El nombre del archivo de verificación debe ser parecido a `google4ddabfacdb4f6795.html`. Por otro lado, los contenidos de Google-site-verification van detrás el nombre del archivo, es decir, quedaría algo como esto: `Google-site-verification: google4ddabfacdb4f6795.html`.

A continuación, debe dejar la ventana del navegador abierta con Google Apps y, en otra ventana, se abre el servidor FTP para entrar en su sitio web y subir el archivo HTML que ha descargado. Este archivo de verificación debe subirse en la carpeta raíz de su sitio web. No puede subir el archivo de verificación a una subcarpeta, ya que, este debe ser visible en <http://dominio.com/nombrearchivo.html> y no en <http://dominio.com/nombrecarpeta/nombrearchivo.html>. Debe saber que el nombre de la carpeta raíz de su sitio variará dependiendo del proveedor de alojamiento que tenga contratado. Sin embargo, los nombres comunes para esta carpeta son `public_html`, `www` o `wwwroot`.

## **Cómo asegurarse de que el archivo HTML está subido correctamente**

Para asegurarse de que ha realizado bien este proceso, lo primero que debe hacer es abrir una ventana del navegador. Posteriormente, ir a la url en donde ha subido el archivo en su sitio web. Esta dirección es, como ha visto anteriormente, el nombre de dominio más el nombre del archivo HTML, es decir, <http://dominio.com/nombrearchivo.html>. Debe tener en cuenta que la parte de

dominio.com debe de ser el nombre de su dominio real y, la parte de nombredarchivo.html, el nombre que le haya puesto al archivo HTML que ha subido.

Por último, si ha subido el archivo de manera correcta, puede ver en el navegador que se muestra el mensaje verificación de sitio de Google, seguido por los caracteres que había en el archivo que ha, previamente, subido.

### **Verificar el dominio completamente**

Para hacer una verificación del dominio completa, debe volver al panel de control de Google Apps donde se muestran las instrucciones de verificación. A continuación, pulse en el botón Verificar que se encuentra en la parte inferior de la página. Al proceder con esta acción, está indicándole a Google que busque el archivo HTML, que ha descargado al principio, en la dirección url donde lo ha subido previamente. Cuando Google haya encontrado ese archivo donde le ha indicado, activará este servicio para su dominio.

#### ***3.1.3. Técnicas CSS***

Llegados a este punto, ya ha comprobado que la estructura de su documento y su contenido son válidos y están verificados. Por lo que, en este apartado lo que va a aprender es a asegurar la validez del formato.

Para cumplir con los estándares, debe haber puesto su código CSS en un formato específico y apartado del código HTML. Esto hace que el estilo de su web esté alojado en una hoja de estilo independiente.

Como vio anteriormente, el W3C ofrece una herramienta capaz de validar código, pero en este caso, en vez de usarla para validar HTML o XHTML, la usa para validar CSS. La herramienta es la [siguiente](https://jigsaw.w3.org/css-validator/)(<https://jigsaw.w3.org/css-validator/>)

Si utiliza esta herramienta tiene tres opciones diferentes para validar una hoja de estilo. Si tiene su hoja de estilo en un servidor accesible desde Internet, tan solo debe facilitar la url de la misma. Otra opción es enviar el archivo de la hoja de estilo al servidor de la herramienta. Por último, la tercera opción es introducir en un recuadro de texto las reglas de validación que quiere que siga la herramienta. Sin importar cuál de las tres opciones utilice para validar su archivo, debe pulsar el botón Check para que se realice la validación. Una vez pulsado, la herramienta mostrará el informe de resultados.

En la herramienta, las dos primeras partes ofrece n una versión extendida donde puede acceder al pulsar los enlaces proporcionados. En esta versión extendida, puede elegir la versión de CSS respecto a la que desea validar su documento. Gracias a esto, puede comprobar si su documento CSS se adapta a la configuración de los dispositivos móviles.

En el caso en que la hoja de estilos presente errores, estos errores se mostrarán en el informe de manera detallada y facilitará la posición donde puede encontrar cada uno de los errores del

documento. Además, aportará una pequeña descripción del error y planteará las posibles soluciones. Por ejemplo, si el informe destaca un error en donde ha utilizado un nombre inválido, para solucionar este error indicará que debe facilitar un tipo de letra genérico al final de la lista de valores de Font-family. El objetivo de validar el código es que obtenga un mensaje de No Error Or Warning Found que aparece en la parte superior de su informe. Este mensaje os indicará que su hoja de estilo es totalmente válida y, esto conlleva que pueda incluir el logotipo correspondiente en su web indicando que su página cumple los requisitos de los estándares.

El paso siguiente a realizar es realizar la verificación de accesibilidad de su web. Para realizar esta verificación, debe realizarla de una manera más manual, ya que, aunque existen servicios que realizan esta verificación, no tienen la suficiente calidad para ofrecer la mejor calidad en su web.

En el supuesto caso en el que quiera validar otro tipo de documentos, el W3C no ofrece de manera directa este tipo de herramientas de validación con la que pueda, por ejemplo, validar documentos XML o WML, por lo tanto, para estos otros tipos de documentos, tendrá que echar mano de herramientas de validación externas. Por ejemplo, puede echar mano de [validator](#), que es una de las herramientas de validación más completas que puede encontrar. Esta herramienta permite validar archivos HTML, WML, XML y XHTML. En cambio, tendría que seguir procesando las hojas de estilo con la herramienta que vio anteriormente.

Cuando acceda a la herramienta, lo primero que debe hacer es facilitar una url o nombre del archivo que quiere verificar. Además, tendrá que elegir el tipo de codificación y el tipo de documento, en el caso de que, así, lo necesite. Por último, puede marcar una serie de opciones, que ofrece la herramienta, antes de pulsar el botón para que comience el proceso de validación del documento.

Utiliza esta herramienta como complemento de la herramienta que ofrece W3C, la cual debe utilizar como opción preferente, para realizar las validaciones pertinentes de documento WML y XML. A continuación, va a ver algunos servicios adicionales de validación:

- Validación de esquemas XML con la herramienta de W3C.
- Validación de XForms, es decir, formularios basados en XML.
- Recursos para validación de la accesibilidad de las webs.

Además, el servicio que ofrece W3C ofrece la revisión del código HTML y CSS por si contienen errores e indica qué tipo de errores son y cómo corregirlos. También, ofrece ejemplos prácticos de los errores más comunes y los daños que causan en el funcionamiento y posicionamiento de su web.

Es obligatoriamente necesario que antes de publicar una página en Internet, compruebe si el código HTML que ha creado es totalmente válido, es decir, que esté libre de errores. En el caso en que encuentre algún error, debe corregirlo inmediatamente para asegurarse de que su página cumpla los requisitos y especificaciones exigidas por los estándares y que, por lo tanto, se muestre de forma correcta en el navegador y en cualquier dispositivo. Además, debe revisar también el código CSS de las hojas de estilo. Todo ello es independiente del software que haya empleado para crear sus webs. Algunos de los errores que presentan las páginas web, pueden perjudicar el posicionamiento de su página web en los buscadores de Internet.

Para finalizar vea los pasos a seguir para la verificación de un sitio web:

**Sistemas ubicados en Internet:**

Para las webs que están alojadas ya en Internet, se recomienda utilizar el servicio de validación de enlaces de [W3C](#).

**Software:**

Si quiere validar su sitio web con un software de escritorio, se recomienda utilizar el programa Xenu. Además, debe saber que los softwares que se utilizan actualmente para crear sitios webs son capaces de manejar los enlaces internos de la propia web de manera controlada. Por ejemplo, un error común de este tipo es que una imagen se vea perfectamente en el lado del desarrollador, pero, sin embargo, no se vea bien en el sitio web final. La razón de que ocurra eso es que la imagen está siendo referida de forma absoluta desde un disco duro local, en vez de desde el directorio donde se encuentran las imágenes del sitio web.

**Sitio en construcción:**

Antes de publicar su sitio web en Internet, debe asegurarse de que no contenga espacios vacíos o que en la web aparezca el mensaje en construcción. No es para nada recomendable que aparezcan espacios en la web, una vez esta esté abierta, con este mensaje. En caso de que sea necesario, es mejor eliminar la zona donde aparece el título y volver a añadirla cuando sea necesario.

**Verificación de metaetiquetas:**

HTML ofrece unos elementos llamados metaetiquetas que posiciona en la parte superior de cada documento. Estas metaetiquetas se utilizan para múltiples fines, pero uno de ellos es que la página sea indexada por los sistemas de búsqueda tales como Google, Bing, Yahoo!, etc. Ya que, las metaetiquetas les aporta a los sistemas de indexación la información mínima para que puedan indexar la página correctamente. Las metaetiquetas obecen a los estándares del W3C, por ello su uso está totalmente regulado. En el caso en que necesite verificar las metaetiquetas para ver si cumplen con los requisitos mínimos requeridos por los buscadores, puede utilizar múltiples herramientas que encuentra en Internet y que permiten realizar la prueba. Además, estas herramientas aportan recomendaciones para mejorar la información que pone a las metaetiquetas.

**Verificación de estándares:**

Tiene claro que puede construir páginas webs con diferentes lenguajes de programación. Sin embargo, todos los sitios que construya deben cumplir con las normas de organización de su propio lenguaje, es decir, la sintaxis. Esto hará que diferentes tipos de software sean capaces de interpretar ese código en diferentes plataformas. La sintaxis de cada lenguaje está estandarizada y puede hacer pruebas para ver si cumple los requisitos exigidos mediante herramientas gratuitas que se encuentran en Internet.

### 3.2. Herramientas de depuración para distintos navegadores

En esta sección, va a aprender cómo depurar código HTML, CSS y JavaScript con las herramientas que ofrece en los diferentes navegadores. Y también, aprenda a revisar el código de páginas ya creadas por usted o por terceros.

Antiguamente, depurar código era una tarea muy tediosa, ya que, normalmente para depurar código JavaScript debía utilizar el comando alert y, mediante el uso de ventanas emergentes, se podían ver los valores que daban como resultado ciertos comportamientos en las aplicaciones.

Sin embargo, actualmente, esta manera de depurar el código ha cambiado. Hoy por hoy tiene múltiples aplicaciones que funcionan como entorno de depuración y que facilitan esta tarea. En cambio, tiene que aprender a cómo sacarles partido a estas herramientas, ya que, hasta ahora no lo ha hecho.

La mayoría de navegadores ofrecen herramientas de depuración y permiten acceder a los elementos web que se hayan descargado. Además, las utilidades genéricas son la consola, elemento/Inspector/HTML, estilos, depurador, profiler que se utiliza para analizar el rendimiento del código JavaScript y las herramientas de red, que permite ver el tráfico de la misma.

Para acceder a estas herramientas del navegador, debe pulsar la tecla F12 de su teclado, si su ordenador es Windows y, en el caso de que su ordenador sea Mac, debe pulsar las teclas Cmd, Option y I. Otra forma de acceder a estas herramientas es accediendo desde los menús superiores o contextuales, es decir, pulsando con el botón derecho del ratón en el elemento que desee revisar y eligiendo la opción inspeccionar elemento. En el navegador Safari, antes de poder utilizar estas herramientas, tendrá que activarlas en la sección de preferencias avanzadas.

Además, puede complementar las herramientas que ofrece en los navegadores con *plug-ins* para aumentar las características de los mismos. Por ejemplo, un *plug-in* muy útil es Firebug. Este tipo de herramientas están en constante desarrollo, por lo que, siempre ofrecen actualizaciones con las que cada vez tendrá una herramienta más pulida. Por otra parte, existen navegadores especializados en el desarrollo web como, por ejemplo, la versión Canary de Chrome o el Mozilla Firefox Developer Edition.

También, puede encontrar el navegador Chromium que es un proyecto de código abierto basado en el código fuente de Google Chrome. El navegador Chromium solo admite códecs como Theodora, códecs WebM y Vorbis para las etiquetas de audio y vídeo en HTML5. En cambio, Google Chrome admite todo lo anterior y los códecs AAC y MP3. Aunque, algunas distribuciones de Linux ponen soporte para los códecs de terceros a sus versiones de Chromium.

Si en la barra de url de Google escribe Chrome://flags, aparecerán una serie de herramientas experimentales las cuales puede activar pulsando la opción Habilitar. Haciendo esto, habrá habilitado estas funciones experimentales para las herramientas de desarrolladores que ofrece Chrome. De todas maneras, puede alternar esta configuración para experimentos individuales. Para ello, tan solo

debe ir al panel de configuración de la herramienta y habilitar el #enable-devtools-experiments. Siempre que realice cambios en la configuración, debe reiniciar el navegador.

Por otro lado, en los navegadores como Firefox, tiene a Firefox Aurora que muestra las últimas actualizaciones que se ponen al navegador Firefox.

En el navegador Safari, también tiene el Webkit Nightly que es una versión de Safari menos estable que en los demás navegadores. En cambio, aunque esta versión de Safari sea menos estable que las demás, puede corregir con ella gran parte de los errores actuales. Esta herramienta se ejecuta junto al navegador Safari.

Por último, en el navegador Internet Explorer, tan solo puede probar sus diferentes versiones. Para los navegadores que no dispongan de barra de desarrollo al pulsar la tecla F12, debe instalar el complemento Explorer developer toolbar, como por ejemplo en este caso con Internet Explorer.

Vea a continuación algunas de las funciones más útiles que puede utilizar desde la consola de depuración:

**Console.log()**. Esta función es extremadamente útil. Se utiliza para ver la salida de datos durante la depuración sin que tenga que utilizar las engorrosas ventanas emergentes de alert. Sin embargo, hay otros métodos con los que puede visualizar la salida de la información de manera más sencilla. Por otra parte, esta función también funcionaría como un printf. Es decir, que podrá imprimir datos y mostrarlos por pantalla. Por ejemplo:

```
Console.log("%s tiene %d años, "Marcos", 18)
```

Esta línea mostrará por pantalla la frase: Marcos tiene 18 años.

- **Console.info()**. Esta función se utiliza para visualizar los mensajes de información.
- **Console.debug()**. Esta función emite mensajes de depuración.
- **Console.warn()**. Esta función emite mensajes de advertencia.
- **Console.error()**. Esta función emite mensajes de error. Además, en esta función también puede utilizar el patrón %c cuando utiliza el segundo argumento como parámetro.

Vea un ejemplo de console.log() cómo reacciona en los diferentes navegadores. Va a emplear el siguiente código (“%c esto es el texto en morado sobre un fondo verde.”;color:purple; background-color:green”)

Si utiliza Firebug, mostrará distintos iconos para estas variaciones, además, mostrará mensajes en fondos de colores.

Por otra parte, si utiliza Firefox va a mostrar un ícono gris a lado de la información, además, mostrará y advertirá mensajes de error sobre elementos que requieren su atención.

Por último, Chrome y Safari mostrarán un ícono informativo, pero, también, mostrará un ícono azul de depuración, un ícono rojo para error y otro amarillo para advertencia.

- **Console.table()**. Esta función se utiliza para mostrar datos de salida de una o varias matrices o de una lista de objetos en formato tabular.
- **Console.asset()**. Se utiliza para probar si las expresiones son verdaderas o falsas.
- **Console.time()**. Se utiliza para añadir un temporizador a la consola.
- **Console.timeEnd**. Es una etiqueta que se utiliza para detener el temporizador.
- **Console.timeStamp**. Esta etiqueta se utiliza para medir cuándo se ejecutó una parte determinada de código.

En el navegador Google Chrome, puede abrir la consola pulsando la tecla F12 aunque este navegador disponga de una pestaña propia para la consola. Por otro lado, el navegador Safari tiene la consola en la parte inferior de las herramientas para desarrolladores. En este navegador, utiliza la tecla Escape para cambiar si se muestra o no la consola. Por último, en el navegador Mozilla Firefox, puede acceder rápidamente a la consola, tan solo, pulsando las teclas Ctrl, Shift y K en Windows y en Mac se accederá pulsando Cmd, Option y K.

Hoy por hoy, debe crear siempre un diseño de su web de manera que sea responsive. Esto quiere decir que su web pueda ser visualizada mediante cualquier dispositivo con cualquier tamaño de pantalla. Incluso las herramientas del navegador están diseñadas para visualizar su web como si la estuviera visualizando en otro dispositivo como, por ejemplo, una Tablet o un móvil. Para ello, va al panel de Configuración de las herramientas para desarrolladores de Chrome. Este panel se encuentra al pulsar en el engranaje que se ubica en la esquina inferior derecha de las herramientas. Debajo de Overrides, entra en la opción de Device Metrics que permitirá modificar la anchura y altura de la pantalla y, también, el tamaño de fuente. Además, puede situar las herramientas de desarrollo en la parte derecha de la página y redimensionarlas al tamaño que quiera.

Por otra parte, las herramientas de desarrollo de Mozilla Firefox son herramientas con las que puede testear los sitios webs con cualquier dimensión. Por ejemplo, si quiere utilizar la herramienta Modo Diseño Sensible, puede encontrarla en la parte superior derecha donde se encuentran las herramientas de desarrollo. Además, este navegador también ofrece una opción donde puede poner la pantalla en horizontal y vertical para poder probar la página en estas dos posiciones.

En lo relativo a las fuentes, el navegador Mozilla Firefox cuenta con un inspector de fuente, el cual permite visualizar la fuente que se está utilizando en el elemento que esté inspeccionando en ese momento. Además, este navegador también va a permitir cambiar el texto predeterminado Abc para que pueda probar la fuente con cualquier otro texto que desee. Por otro lado, en la parte inferior de la lista de fuentes puede encontrar un botón que utiliza para visualizar todas las fuentes que hay en la página. Esto es una forma fácil y rápida de que pueda ver de un vistazo todas las fuentes que se utilizan en la página. Por último, si utiliza el *plug-in* Firebug va a ofrecer información sobre cualquier declaración de tipo de letra que aparezca en código CSS.

En caso de que necesite modificar los márgenes, el relleno, la altura o la anchura de una web usando código CSS, puede utilizar las teclas de cursor para disminuir o aumentar el tamaño. Las teclas de

---

cursor que utiliza serán la de arriba y abajo, eso conllevará que aumente o disminuya una unidad de tamaño. En cambio, si quiere aumentar o disminuir en unidades de diez en diez, debe utilizar las mismas teclas de cursor, pero pulsando al mismo tiempo la tecla Shift. Además, también puede disminuir o aumentar 0,1 unidad pulsando las teclas cursor y la tecla ALT. Por último, puede aumentar o disminuir unidades de cien en cien si utiliza las teclas Repág o Avpág y la tecla Shift.

Estos atajos de teclado son muy útiles cuando no sabe qué tamaño exacto es el que quiere poner, por lo que al utilizar estos atajos va a ser más fácil y rápido hacer comprobaciones del tamaño. Además, permiten corregir las distorsiones CSS en tiempo real antes de subir la página a Internet.

## RECURSO MULTIMEDIA

### 3.2.1. Utilidades para HTML

Las herramientas que encuentra al apretar la tecla F12 nos van a ayudar a encontrar y resolver errores en el código HTML y CSS, cosa que se haría mucho más difícil de realizar en el código de origen. Esto ocurre porque cuando el navegador va interpretando el código el código HTML y CSS se va mostrando en un árbol del DOM. De esta forma, los cambios dinámicos que haya efectuado, serán más fáciles de depurar.

#### Uso de pestañas en HTML

En HTML puede utilizar la vista de pestaña que sirve para mostrar en forma de árbol el marcado dinámico de la web. Esta es la manera como representaría el navegador la información de la web en memoria. Si se realizan cambios en la web, las herramientas F12 reflejan cómo funciona el DOM, es por ello que deben abrirse y actualizarse. Puede navegar por el árbol DOM mediante el uso del ratón o del teclado, además, puede cambiar los valores y atributos de los elementos.

Si mientras está navegando por el árbol quiere pasar a un elemento determinado en la página, tan solo debe pulsar las teclas Ctrl y B o pulsar el botón Seleccionar elemento. Además, cuando estén navegando por la página, los elementos se resaltan cuando pasa el puntero del ratón por encima de ellos. Cuando pulsa encima de un elemento, lo que ocurre en el árbol DOM es que ese nodo se resalta en la vista HTML. Al seleccionar un elemento en la pestaña HTML, puede filtrar la vista con los

elementos que ha seleccionado donde se mostrarán los estilos CSS asociados a ellos o, en caso de no tener estilos, se mostrarán los elementos en sí. Para ello, debe ir al menú Ver y seleccionar la opción código fuente.

Por otro lado, si pulsa el botón Origen del elemento con estilo, se mostrará únicamente el código fuente y el contenido del elemento que haya seleccionado. Además, mostrará también en una ventana el código CSS que está asociado al mismo. Esta opción, ayudará a focalizarse en el código fuente del elemento o elementos que haya seleccionado. Antes de ejecutar esta acción debe haber seleccionado con la vista de pestaña los elementos del cuerpo HTML en el árbol del DOM. Para seleccionar los elementos, tan solo, debe pulsar en el botón Seleccionar un elemento o pulsar el elemento del propio árbol del DOM. Antes de utilizar el botón Seleccionar un elemento, debe actualizar primero la vista.

En caso de que quiera contraer o expandir las propiedades de un elemento, tiene que pulsar en el recuadro que tiene marcado un signo menos o un signo más. Debe tener en cuenta que puede modificar de manera inmediata todos los elementos de las vistas tanto CSS como HTML. Además, puede activar o desactivar las reglas de estilo y los atributos de las mismas. Para ello, se selecciona la casilla que se encuentra al lado del nombre del propio elemento a editar.

### **Visualización del código HTML dinámico**

Hoy por hoy casi todos los sitios web utilizan código dinámico. Los scripts que se utilizan en el lado del cliente, normalmente, se utilizan para generar el código HTML que se muestra al usuario final. Aunque, aún se utiliza código HTML estático, este solo se utiliza para iniciar eventos en la página. Como ya vio anteriormente, con las herramientas F12 es más fácil y rápido encontrar errores y solucionarlos, ya que, estas herramientas muestran el código CSS y HTML como procesados por el navegador y no como origen de HTML.

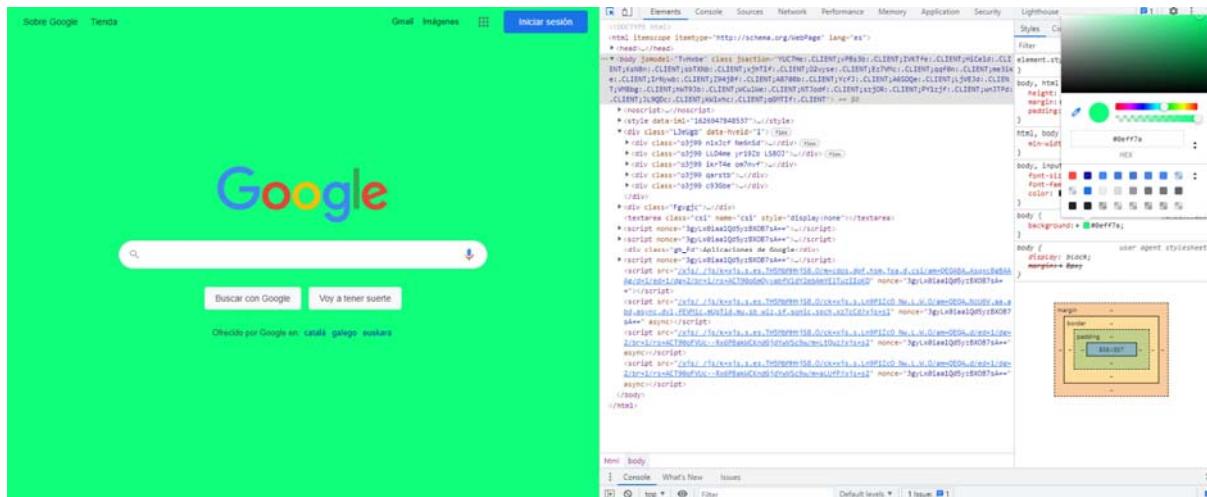
Por otro lado, en la pestaña de HTML no puede realizar un control de las modificaciones de forma automática, pero sí puede mostrar el estado actual del DOM. Si al cambiar un atributo o un valor no se muestra de manera instantánea, debe actualizar la vista en la pestaña de HTML pulsando la tecla F5.

### **Modificar valor y añadir atributos**

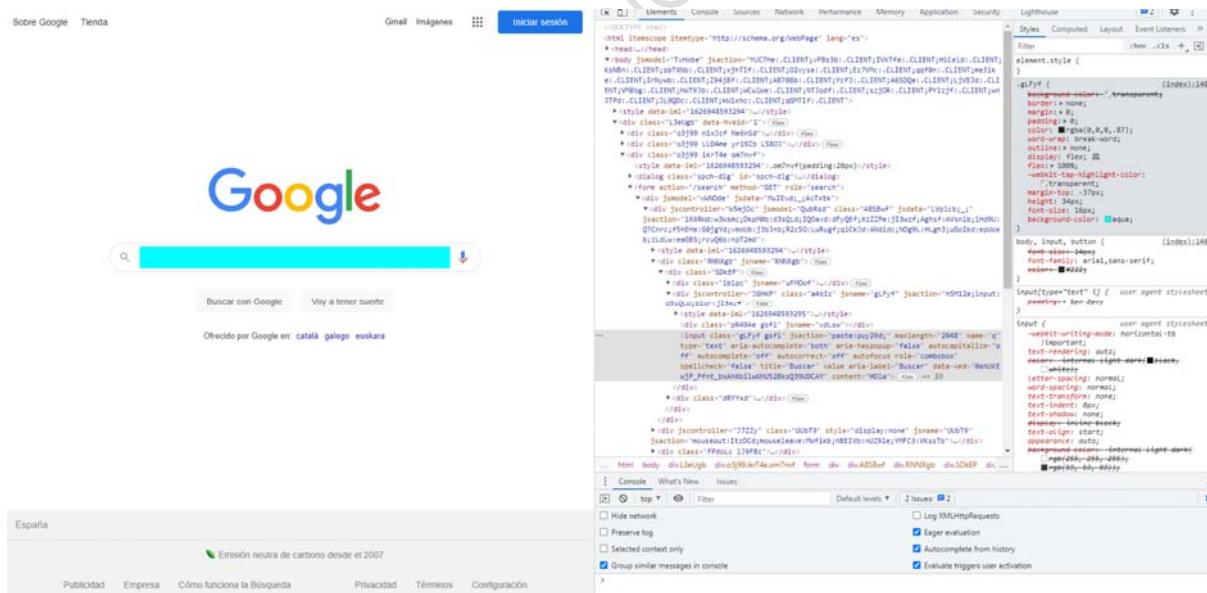
Para realizar esta acción, debe utilizar las herramientas F12, ya que, permiten modificar un valor de cualquier atributo o propiedad. Para ello, va a la pestaña HTML y selecciona un elemento en el árbol DOM, también puede utilizar el botón Selecciona un elemento. A continuación, encuentra en el panel derecho, una lista de propiedades, pulsa la propiedad que quiere modificar y esta se verá resaltada y podrá ser modificada.

Vea un ejemplo de lo que ha explicado en el párrafo anterior. Para ello, va a abrir una página cualquiera en su navegador. A continuación, presiona la tecla F12 para abrir las herramientas de desarrollo. Pulsa el botón Seleccionar un elemento y, a continuación, pulsa uno de los títulos de la web. Por otra parte, en el panel de propiedades, pulsa en el atributo de color de fondo del body. Una

vez seleccionado el atributo, va a cambiar el color del mismo cambiando el nombre del color, por ejemplo, verde y presiona la tecla Enter. Como resultado, el color del elemento habrá cambiado de manera instantánea. Vea el ejemplo:



En caso de que quiera agregar un atributo, tan solo, debe presionar con el botón secundario del ratón sobre un elemento del panel izquierdo, ya sea, de la pestaña CSS o HTML, y debe hacer pulsar en la opción Agregar atributo. Por ejemplo, va a ponerle un color de fondo a la barra buscadora de Google. Para ello, va a añadir un atributo en el CSS. Vea el ejemplo:



Por otro lado, si quiere eliminar un atributo, ya sea uno que haya creado o uno que ya hubiera creado, debe pulsar en el atributo que quiere eliminar en el panel derecho, ya sea, el panel de HTML o CSS y, presionar la opción Eliminar. Si quiere que vuelvan a estar los atributos de la página original que haya eliminado, tan solo, debe recargar la página. Como es lógico, debe volver a aplicar los atributos creados para que puedan ser visualizados en la página de nuevo, ya que, al recargar, éstos se eliminan.

En caso de que en vez de eliminar el atributo solo quiera desactivarlo temporalmente, solo debe desactivar la casilla que se encuentra junto al atributo a desactivar en el panel derecho. Vea el ejemplo:

The screenshot shows the Chrome DevTools Styles tab. It displays two CSS rule blocks. The first block is for the element `element.style` and the second is for the selector `body, input, button`. In the first block, there is a property `background-color` with a value of `transparent`. To the left of this property, there is a checkbox that is checked. Below this, there is another checkbox that is unchecked, indicating that the original value (`aqua`) is still present. The second block contains properties for font size, family, and color.

```
element.style {
}
.gLFyf { (index):140
 background-color: transparent;
 border: none;
 margin: 0;
 padding: 0;
 color: #rgba(0,0,0,.87);
 word-wrap: break-word;
 outline: none;
 display: flex;
 flex: 100%;
 -webkit-tap-highlight-color: transparent;
 margin-top: -37px;
 height: 34px;
 font-size: 16px;
 background-color: aqua;
}
body, input, button { (index):140
 font-size: 14px;
 font-family: arial,sans-serif;
 color: #222;
}
```

## Herramientas y vistas de la pestaña HTML

Al seleccionar un elemento en el panel izquierdo, puede ver y cambiar estilos en el panel derecho. Además, también puede modificar los atributos de los elementos y el diseño del modelo de cuadros. Debe tener en cuenta que las modificaciones que realice de esta manera, no son permanentes, por lo tanto, al cerrar o actualizar el navegador, las modificaciones desaparecerán. En cambio, puede guardar el código HTML si pulsa el botón Guardar.

Cuando selecciona un elemento en el que observa que se aplican varias reglas CSS, se debe a que estas reglas se están mostrando respecto a su especificidad requerida con la especificación del código CSS. Por lo tanto, la regla que se encuentre en la parte superior, será la que primero se aplique al elemento seleccionado, y la regla que se encuentra en la parte inferior definirá las propiedades del estilo del elemento. Como ya sabe, puede modificar los valores de estas reglas. Para ello, pulsa en un valor, escribe el nuevo valor y presiona la tecla Enter. Como puede observar, la modificación se hará efectiva de manera inmediata. La información que puede encontrar en la pestaña Rastrear y la pestaña Estilos es la misma. En cambio, en los tipos de propiedad que encuentra en Rastrear cómo las propiedades se agrupan de manera distinta. Estas propiedades se

mostrarán en orden alfabético y las reglas, como dijo anteriormente, se van a ordenar según su especificidad.

Vea a continuación las partes y vistas de las herramientas HTML:

- **Diseño.** Aquí se muestra el modelo de cuadros del elemento que haya seleccionado. Si pulsa en el diagrama puede cambiar cualquier valor de la vista de Diseño. Esta pestaña estará inactiva para los elementos SVG.
- **Atributos.** En esta pestaña se muestran los atributos como, también, los identificadores de los elementos seleccionados. Puede modificar o eliminar cualquier atributo.
- **Rastrear estilos.** En esta pestaña se muestra la misma información que en la pestaña Estilo. En cambio, se agrupa por propiedades y no por reglas.
- **Estilo.** La pestaña estilo muestra las reglas y los estilos del elemento que haya seleccionado anteriormente en la vista árbol. Esta organizada por reglas y, además, se muestran todos los atributos tanto heredados como desactivados.

#### **Presionar con el botón secundario del ratón en el Menú de la pestaña HTML**

Como ya vio anteriormente, puede pulsar con el botón derecho del ratón encima de un elemento que se encuentre en el panel izquierdo de la pestaña HTML. En la siguiente tabla, se muestran las opciones que puede utilizar en la pestaña HTML:

Elemento del menú	Acción
Añadir atributo	Añadir nuevos atributos a una etiqueta o a un elemento
Copiar	Se utiliza para copiar los atributos y la etiqueta en el portapapeles
Copiar innerHTML	Se utiliza para copiar el contenido del innerHTML de un elemento
Copiar outerHTML	Es lo mismo que innerHTML, pero el contenido copiado es el del outerHTML

Por último, si estando en la pestaña de HTML, pulse en el nodo secundario de un elemento, por ejemplo, el elemento h1, puede ser que no obtenga todas las opciones. Si ocurre que no puede encontrar la opción que está buscando, debe buscar, entonces, el elemento principal.

#### **3.2.2. Utilidades para JavaScripts**

En este apartado, va a aprender qué es HTML TIDY. Este elemento ayuda a corregir los errores que muestra todos los servicios de validación que ha visto anteriormente. Corregir estos errores sin HTML TIDY sería una tarea muy engorrosa que debería realizar de forma manual. HTML TIDY es un programa gratuito de código abierto que ofrece el W3C. Este programa se encuentra disponible en diversos formatos para que pueda utilizarlo en diferentes Sistemas Operativos y, para que pueda corregir errores de múltiples lenguajes de programación.

Este programa puede identificar la mayoría de los errores comunes que pueden suceder en el código HTML, XHTML y XML, al igual que las herramientas de validación que ha visto anteriormente. Sin embargo, lo que diferencia esas herramientas de este programa es que HTML TIDY, además de identificar errores, es capaz de corregirlos. Con este programa, también, puede convertir un documento HTML en un documento XHTML, modificar las líneas del documento a un número máximo de columnas, añadir sangrías a los elementos, etc. Con este programa, también, puede realizar comprobaciones de los distintos niveles de accesibilidad determinados por el W3C.

Puede obtener el programa HTML TIDY en SourceForge, donde debe elegir la versión que necesite dependiendo de su Sistema Operativo. Una vez tenga el programa, debe llamarlo desde la línea de comando especificándole qué opciones quiere aplicar e indicarle qué archivo debe procesar. También, en la consola de comandos, puede escribir el comando Tidy-help para que se muestre una lista con todos los comandos disponibles. La interfaz de este programa va a ser siempre igual, sin importar qué Sistema Operativo esté utilizando.

Si utiliza Tidy HTML desde el propio ordenador y lo ejecute desde la línea de comando, tendrá la ventaja de que puede modificar los archivos que quiere procesar. De esta manera, puede hacer correcciones sin tener que dar ningún paso extra. Sin embargo, también puede utilizar este programa desde Internet sin necesidad de tenerlo en su ordenador. Esta opción permitirá validar la página que desee, tan solo, añadiendo la url. Además, también puede validarla introduciendo directamente el código HTML en la caja de texto o enviar el archivo al servidor de la aplicación. Una vez que haya decidido qué opción de validación escoger, debe elegir las opciones de cómo va a validar el programa su archivo. Después, tan solo debe pulsar el vínculo Tidy! para obtener el informe y el archivo corregido. Tendrá la opción de recuperar el archivo original.

Una de las ventajas de utilizar la aplicación en la interfaz web es que no debe aprender los comandos para poder manejar Tidy. En cambio, el inconveniente es que en la versión web no se utiliza la versión más actualizada de Tidy, por lo que, las correcciones de sus documentos no se harán de manera tan efectiva como en la versión de escritorio.

Por otro lado, una página web con un buen diseño CSS y un marcado semántico de calidad es casi perfecta. Sin embargo, para que esa página sea de utilidad debe tener una buena interactividad con el usuario. A esto lo denomina capa de comportamientos y, normalmente, se crea mediante el uso de lenguaje JavaScript.

El lenguaje JavaScript es un lenguaje de programación basado en objetos que se creó en 1995 para Netscape. Este lenguaje empezó llamándose Mocha, posteriormente pasó a llamarse Livescript y concluyó llamándose JavaScript en su estreno en el navegador Netscape versión beta 2.0.

Debe tener muy en cuenta que cuando se refiere al lenguaje de programación JavaScript no está refiriendo al lenguaje de programación Java, aunque ambos lenguajes tienen su base en el lenguaje C, por lo que comparten una sintaxis similar. Sin embargo, es en lo único que se parecen. La similitud entre ambos nombres radica en que el lenguaje de programación Java estaba siendo muy reconocido en el mundo de la informática cuando Netscape estaba dándole las últimas pinceladas al lenguaje LiveScript, por lo que, la compañía Netscape aprovechó el momento para llamar a su lenguaje

---

JavaScript y darle mayor publicidad. En cambio, este hecho en vez de darle popularidad a JavaScript, lo que hizo es crear confusión entre ambos lenguajes. Finalmente, JavaScript se reguló y estandarizó en el año 1996 cuando cayó su mantenimiento y desarrollo en manos de la ECMA. En 1999, llegó la versión 1.5 de JavaScript que corresponde a la versión ECMA-262 Edición 3. Por esta razón, muchas veces denomina JavaScript como ECMAScript.

Por otra parte, Microsoft también creó un lenguaje llamado Visual Basic Script que estaba basado en el lenguaje Visual Basic, el gran inconveniente de utilizar este lenguaje es que solo tiene soporte en el navegador Internet Explorer, por lo que si quiere crear una página web será una pésima idea utilizar este lenguaje, ya que, solo podrá ser interpretada por un solo navegador. Además, Microsoft también intentó crear su propio lenguaje JavaScript, llamado JScript que soportaba la mayoría de las funciones predeterminadas de JavaScript. Sin embargo, al igual que Visual Basic script, solo funcionaba con el navegador Internet Explorer, por lo que tampoco era una buena apuesta crear una web con ese lenguaje.

El lenguaje Javascript se puede utilizar para más fines que ser interpretado por un navegador. Por ejemplo, con JavaScript puede controlar la plataforma de Mozilla, este lenguaje también está involucrado en el Dashboard de Widgets de Mac OS X 10.4 y, por último, es un lenguaje que también puede controlar en los documentos PDF. Debe tener en cuenta que JavaScript es un lenguaje pensado para estar en el lado del cliente, es decir, es un lenguaje pensado para que se ejecute en el ordenador de los usuarios y no en el servidor. Esto es importante tenerlo en cuenta porque debe ser conscientes que la ejecución de este lenguaje estará involucrada con la capacidad que tenga el navegador que lo está interpretando.

A continuación, va a detallar más de cerca la mejor manera de utilizar JavaScript. JavaScript es una herramienta más que tiene tanto sus cosas buenas, como sus cosas malas. Esta herramienta ofrece múltiples posibilidades de uso, por ejemplo, en las ventanas emergentes, en las páginas de inicio de cualquier web, etc. Como desarrolladores web siempre debe tener en cuenta cómo van a influir sus decisiones a la hora de crear el código para una página web, ya que, lo que debe primar es que la experiencia de usuario sea lo más positiva posible. Por lo tanto, al crear las webs no debe centrarse solamente en que funcionen con JavaScript, sino que la web pueda funcionar de manera sólida sin utilizar JavaScript. Esto se debe a que muchos usuarios utilizan navegadores que no pueden interpretar JavaScript y, otros tantos usuarios, desactivan el uso de JavaScript en sus navegadores.



**TOME NOTA**

*Muchas empresas desactivan JavaScript de sus navegadores por motivos de seguridad, ya que, muchas veces el código dañino que dificulta la navegación web está creado con este lenguaje de programación.*

Otro punto donde también debe hacer hincapié es que, debido a razones de accesibilidad, es importante que su página pueda funcionar correctamente con el uso de JavaScript desactivado.

Puede implementar JavaScript en una sola página o, por el contrario, implementarlo en todo el sitio web. Al igual que puede hacer con el CSS, puede incrustar JavaScript en el propio documento HTML o, puede crear un archivo .js externo e importarlo a través de un enlace a su documento HTML. Ambos métodos los lleva a cabo con el elemento script. Vea el ejemplo siguiente:

```
<script type="text/javascript">
//<![CDATA[
Inserte aquí el código JavaScript
//]]>
</script>
```

Como puede observar el elemento script es el que utiliza para crear el bloque donde insertar los scripts. Además, también puede ver que este bloque se ajusta al tipo MIME mediante el uso del atributo type. Por otro lado, la parte que envuelve el //<![CDATA//]]&gt;, significa que todo lo que haya en su interior es un comentario. De esta manera, cuando se ejecute el script, se ignorará lo que haya dentro de esta sección.</p>

Sin embargo, el método más profesional de utilizar JavaScript es utilizarlo de manera externa al archivo principal y añadiendo un enlace a ese código JavaScript. De esta manera, puede reutilizar ese código para todas las páginas que lo necesite y así, puede resumir el número de comentarios que debe utilizar para indicar dónde van los scripts y para qué se utilizarían. Para añadir el script de manera externa lo haría de la siguiente manera:

```
<script type="text/javascript" src="misScripts.js"></script>
```

Como puede observar, en este ejemplo no aparecerá el script como tal, sino que el script se hallará en otro archivo y ese archivo lo enlaza al archivo HTML principal mediante el uso de esta línea de código. El archivo del script lo habría llamado misScripts.js.

En definitiva, puede enlazar tantos archivos script como desee, e incluso puede utilizar scripts incrustados y scripts externos al mismo tiempo. La diferencia entre los scripts externos e incrustados es que los archivos que contengan los scripts externos estarán ubicados en el servidor, mientras que los scripts incrustados, viajarán con el archivo HTML hasta el usuario para ser interpretados por el navegador.

Si utiliza scripts incrustados, se deberán ubicar siempre en la cabecera del archivo HTML o XHTML. Esto se hace de esta manera por convención y para facilitar el mantenimiento de este tipo de scripts. Sin embargo, si ubica el script dentro del body funcionaría igualmente.

El lenguaje de script JavaScript es un lenguaje bastante sencillo de aprender. Es muy parecido a lenguajes como ActionScript que es un lenguaje que permite el funcionamiento de Flash y pertenece

a la misma empresa que JavaScript, ECMA. También, se parece a PHP, aunque este lenguaje se utiliza más en la programación del lado del servidor. Lo que hace que JavaScript sea un lenguaje sencillo de aprender es que es un lenguaje poco tipado, es decir, tiene mucha flexibilidad a la hora de nombrar y declarar variables entre otras cosas.

Debe tener en cuenta que en este lenguaje antes de poder utilizar una variable debe declararla antes. Para declarar una variable, utiliza la palabra reservada var y a continuación pondrá el nombre de la variable a declarar. Los nombres de las variables pueden contener letras, números y algunos caracteres no alfanuméricos. Sin embargo, debe evitar el uso de operadores aritméticos y comillas para nombrar sus variables. Además, tendrá que elegir nombres de variables que no entren en conflicto con las palabras reservadas del propio lenguaje.

En JavaScript, las variables pueden albergar diversos tipos de datos, los tipos de datos en sí se pueden dividir en dos categorías, datos escalares y datos en array. Las variables de tipo escalar solo pueden contener un valor en su interior. Ese valor puede ser un número, un valor booleano o una cadena de caracteres. En cambio, las variables de tipo array sí pueden albergar varios valores.

A continuación, vea los tipos de datos que puede almacenar una variable:

- **Cadenas.** Este tipo de datos son cadenas de caracteres, es decir, datos de tipo texto en general. Para asignar este tipo de dato a una variable, debe escribir ese dato entre comillas ya sean dobles o simples. De esta manera, delimitará el principio y el final de la cadena de caracteres.
- **Números.** Los datos numéricos pueden ser números enteros o números decimales, donde en los decimales hay diversos tipos de decimales. Para asignar un número a una variable tan solo debe escribir el valor sin tener que utilizar ningún tipo de comillas. Para escribir números decimales utiliza el punto y no la coma, es decir, en vez de escribir 1,50; escribe 1.50.
- **Booleanos.** Este tipo de variables también se denominan como variables de tipo lógico, ya que, estas variables solo almacenan dos tipos de datos, 0 o 1, que también puede ser representado como true o false. Este tipo de variables se utilizan para realizar controles lógicos en el comportamiento, por ejemplo, de una función. Si al realizar una acción puede realizarla de diferentes formas, tendrá que programar todas las formas y sus respectivas soluciones, ahí es donde funcionan este tipo de variables.
- **Arrays.** Puede denominar a los arrays como matrices o vectores. Sin embargo, también puede llamarlo arrays aunque sea en español, cualquier persona que sepa de programación va a entender si utiliza esta palabra.

Los arrays son conjuntos de variables que pueden funcionar todas a la vez o una a una. Por ejemplo, si necesita crear una función que utilice los días de la semana, podría crear siete variables para cada día de la semana. En cambio, si utiliza un array puede almacenar esos siete días en una sola variable y utilizarlos todos o los que necesita para la aplicación de la función.

Vea los tipos de operadores que existen en JavaScript. Puede encontrarse dos tipos de operadores:

## Operadores aritméticos

- **Suma (+).** Este operador se utiliza para sumar dos o más valores numéricos o para concatenar cadenas de caracteres entre sí, ya sean, caracteres numéricos o textuales. Vea un ejemplo:

```
Var variable1= 11, variable2="Buenas", variable3="tardes", variable4=20;
```

```
Document.write(variable1+variable4) /*Resultado 31*/
```

```
Document.write(variable2+variable3) /*Resultado Buenas tardes*/
```

```
Document.write (variable1+variable3) /*resultado 11 días*/
```

- **Resta (-).** Este operador se utiliza para restar valores numéricos. Además, se puede utilizar para cambiar el signo a un operador numérico. Vea un ejemplo:

```
Var num1=11, num2=9, cont=0;
```

```
cont=num1-num2; /*cont alberga el valor 2*/
```

```
cont=-cont /*cont alberga ahora el valor -2*/
```

**Producto (\*) y cociente (/).** Estos operadores se utilizan para multiplicar y dividir valores numéricos. Vea un ejemplo:

```
Var num=40, num2=2, div, mul;
```

```
Div= num1/num2 /*div alberga 20*/
```

```
Mul= num1*num2 /*mul alberga 80*/
```

- **Resto o módulo (%).** Este operador se utiliza para calcular el resto de una división. Vea el ejemplo:

```
Var num1= 50, num2=4, resto;
```

```
Resto= num1%num2; /*resto alberga 1*/
```

**Decremento (--) e incremento (++)**. Utiliza estos operadores para incrementar y decrementar unidades en una variable. Si quiere que este incremento o decremento sea prioritario a cualquier operación, debe posicionarlo antes de la variable. Vea el ejemplo:

```
Var num1=8, cont;
```

```
Cont= ++num1; /*Primero cont almacena el valor de 9 y después num1 pasa a valer 9*/
```

Cont = num1++; /\*Primero num1 pasa a valer 9 y después cont toma el valor de 9\*/

- **Compuestos.** Los operadores compuestos son los mismos operadores de suma, resta, multiplicación y división que se asocian al operador de asignación (=) para cambiar el valor de una variable. Es decir, permiten resumir código y asignar y operar una variable al mismo tiempo. Vea un ejemplo:

Var num=10, text="Bienvenido";

Num+=5; /\*num obtiene el valor de 15\*/

Text+=s'; /\*text obtiene el valor de 'Bienvenidos'\*/

Num\*=10; /\*num obtiene el valor de 100\*/

### Operadores de comparación

Operador	Descripción	Ejemplo
<b>Igual (==)</b>	Devuelve true si los valores son iguales	4==4 true 4=="4" true 4==5 false
<b>Distinto (!=)</b>	Devuelve true si los valores son distintos	4!=5 true 4!="5" true 4!=4 false
<b>Igual estricto (===)</b>	Devuelve true si los valores son del mismo tipo y del mismo valor	4==4 true 4=="4" false
<b>Distinto estricto (!===)</b>	Devuelve true si los valores son de diferente tipo y diferente valor.	4!=="5" true 4!==5 false
<b>Mayor que (&gt;)</b>	Devuelve true si el valor es mayor que el valor con el que se compara	5>4 true 5>9 false
<b>Mayor o igual que (&gt;=)</b>	Devuelve true si el valor es mayor o igual que el valor con el que se compara	5>=5 true 5>=4 true 5>=6 false
<b>Menor que (&lt;)</b>	Devuelve true si el valor es menor que el valor con el que se compara	5<9 true 5<4 false
<b>Menor o igual que</b>	Devuelve true si el valor es menor o igual que el valor con el que se compara	5<=5 true 5<=4 true 5<=9 false

En JavaScript, puede encontrar diferentes estructuras de control como, por ejemplo, las condicionales, los bucles, las funciones y los intercambiadores.

En JavaScript, también puede encontrar la función alert que es una función muy útil para depurar el código JavaScript. Para utilizarla lo hará de la siguiente manera:

```
Alert('Mensaje');
```

Esta función la puede utilizar cuando requira obtener un poco de información de cómo se está ejecutando y comportando el código, o si necesita alertar al usuario de algo que haya sucedido. Sin embargo, utilizar esta función de manera continua no es buena idea, ya que, puede abrumar al usuario si cada poco tiempo se le abre una ventana nueva mostrando un alert. Para ello, puede hacer uso de otras herramientas que permiten mostrar información en pantalla sin tener que echar mano de la función alert(). Para estos casos en que tenga que mostrar información en pantalla, puede hacer uso de la herramienta jsTrace o fvLogger que es muy parecida a la consola de JavaScript que es la misma que puede ver en los navegadores generales.

Por otro lado, JavaScript permite crear funciones que son trozos de código que puede reutilizar una y otra vez sin necesidad de tener que volver a escribir el mismo código de nuevo. Es por ello que, las funciones son esenciales para cualquier programador. Vea un ejemplo:

```
Function nombreFuncion (argumento){
 Declaraciones;
}
```

Function es una palabra reservada que permite designar un bloque de código como función. Por otra parte, los argumentos pueden ser desde una variable hasta todas las que necesita para conseguir la acción que desea. Además, dentro de los argumentos también puede hallar funciones anidadas.

Las funciones no tienen por qué devolver un valor. También, pueden prescindir de un nombre. A este tipo de funciones, se les llama funciones anónimas que suelen ser asignadas a variables y ser utilizadas como objetos.

Al utilizar y crear variables en JavaScript, debe tener muy presente el concepto de alcance de variable, es decir, el “espacio” que alcanza la variable para ser de utilidad. Es por ello que existen variables locales y variables globales.

Las variables globales son las que se declaran y se inicializan fuera de cualquier tipo de función y son alcanzables desde cualquier parte del programa. En cambio, las variables de tipo local son variables que se declaran dentro de una función y solo se puede acceder a ella desde esa función, es decir, no se puede acceder a ella desde ninguna otra parte. Es por ello que, la palabra var es una palabra clave muy importante para determinar el ámbito de las variables.

Como ya sabe, JavaScript es un lenguaje basado en objetos y, por lo tanto, muchos de sus elementos son objetos. En JavaScript tiene objetos nativos que ya ha visto como, por ejemplo, Array y Function. Otros objetos nativos que puede encontrar son Element, Math y Date.

Además, también puede crear sus propios objetos. Los objetos en sí son un conjunto de datos que puede dividir entre propiedades y métodos. Las propiedades son lo que antes se denominaba valores y los métodos serían las funciones. La esencia de un objeto es que tiene acceso a sus propias propiedades y métodos.

Objeto	Método o propiedad	Descripción
<b>Array</b>	Length	Devuelve o establece los valores de un array.
	Concat()	Une dos o más arrays y devuelve el resultado.
	Join()	Junta todos los elementos en una cadena separados por un limitador específico.
	Pop()	Extrae y devuelve el último elemento de un array.
	Push()	Pone uno o más elementos al final de un array y devuelve la longitud actual.
	Reverse()	Invierte el orden de los elementos.
	Shift()	Extrae y devuelve el primer elemento de un array.
	Slice()	Devuelve los elementos seleccionados de un array.
	Sort()	Ordena los elementos de un array.
	Splice()	Extrae y pone nuevos elementos a un array.
<b>Date</b>	getDate()	Toma la fecha y hora actual.
	getDay()	Toma el día del mes actual.
	getFullYear()	Toma el día de la semana actual.
	getMonth()	Toma el año con cuatro dígitos.
<b>Math</b>	Abs (x)	Devuelve el valor absoluto de un número.
	Ceil (x)	Devuelve el valor de un número redondeando siempre hacia arriba.
	Floor (x)	Devuelve el valor de un número redondeando hacia abajo.
	Max(x,y)	Devuelve el número con valor más alto de x y de y.
	Min (x,y)	Devuelve el número con valor más bajo de x y de y.
	Random ()	Devuelve un número aleatorio entre 0 y 1. Puede configurarla

		para que dé un número aleatorio entre dos números que elija.
	Round (x)	Redondea un número hasta el número más cercano.
<b>String</b>	Length()	Devuelve el número de caracteres de una cadena.
	Concat()	Une dos o más cadenas.
	indexOf()	Devuelve la posición de la primera ocurrencia de un valor de cadena.
	lastIndexOf()	Devuelve la posición de los últimos caracteres de cadena especificado.
	Match()	Busca un valor determinado en una cadena.
	Replace()	Sustituye unos caracteres por otros en una cadena.
	Slice()	Extrae una parte de una cadena y la devuelve a una cadena nueva
	Split()	Divide una cadena en un array de cadenas.
	Substring()	Extrae caracteres de una cadena entre dos índices especificados.
	toLowerCase()	Convierte una cadena en minúsculas.
	toUpperCase()	Convierte una cadena en mayúsculas.

### 3.2.2.1. Depurar JavaScript con Firebug

Cuando programas es inevitable cometer algún que otro error, es por ello que es imprescindible aprender a depurar bien el código y, por supuesto, tener una herramienta que nos ayude a ello, ya que, de manera manual es un engorro depurar el código. Para ello, utiliza una herramienta llamada Firebug que es un *plug-in* de Mozilla Firefox.

Al ser un *plug-in* de Firefox, debe instalarlo antes de poder utilizarlo. Para instalarlo, siga esta [guía](#) paso por paso. Una vez haya instalado el *plug-in*, abrirá en el navegador el archivo que quiere depurar, y pulsa en el ícono de Firebug que aparecerá en la esquina superior derecha de la ventana del navegador. Otra opción es presionar directamente la tecla F12 para acceder al *plug-in*.

A continuación, aparecerá en la parte inferior, la ventana del programa. Como cuando ha cargado el archivo no se estaba ejecutando Firebug, debe presionar F5 para recargar la página y que vuelva a cargar el archivo, esta vez en Firebug. De manera inmediata, verá las líneas de script en la parte inferior izquierda de la ventana.

Llegados a este punto el objetivo no es que se ejecute el código JavaScript, sino que la ejecución se detenga en una línea para que pueda comenzar a analizar el script. Por ello, debe poner un punto de interrupción en el código de la siguiente forma:

Primero, lleva el puntero del ratón a la parte izquierda de la línea donde quiere posicionar el punto de interrupción. Despues, pulse en ese espacio en blanco y verá que aparece un punto marrón que indica dónde se detendrá la ejecución del programa.

Una vez haya establecido el punto de interrupción, debe refrescar nuevamente la página pulsando, para ello, la tecla F5. De este modo, fuerza al programa a ejecutar de nuevo el script el cual se detendrá en el punto de interrupción que ha establecido anteriormente. Si observa, cuando el script se vuelve a ejecutar, aparece, encima del punto de interrupción que ha establecido, un triángulo de color amarillo. Y, además, aparecerá un color de fondo en la siguiente línea a ejecutar indicando que es la próxima línea que se ejecutaría si el programa avanzase. Por otra parte, en el lado derecho, puede observar que aparecen las variables declaradas, pero aún no tienen ningún valor asignado.

El próximo paso es revisar el código del script paso a paso. Para ello, pulsa la tecla F11. Cada vez que presione esta tecla, verá que se resaltará la siguiente línea de código a ejecutar, es decir, que está ejecutando el código, línea por línea, para analizar qué sucede exactamente en cada momento. También, tiene la opción de realizar el mismo proceso con la herramienta F12.

Las herramientas F12 van a permitir depurar el código JavaScript de manera rápida y eficiente sin tener que salir del navegador web. Estas herramientas están presentes en todos los navegadores más conocidos. Las herramientas F12 ofrecen n herramientas de depuración tales como poder visualizar las variables locales, permitir establecer puntos de interrupción, una consola para mensajes y la ejecución instantánea del código. Además, va a ver una lista donde aparecen la mayoría de tareas que permiten realizar estas herramientas:

- Usar la consola para buscar errores de sintaxis y otro tipo de errores del código.
- Depurar varios scripts a la vez.
- Establecer puntos de interrupción condicionales.
- Búsqueda de la pila de llamadas.
- Iniciar y detener el depurador.
- Navegar por el código.
- Interrumpir la ejecución del código.
- Inspeccionar las variables dentro de las pestanas Variables locales y Ver.
- Mejorar los scripts.
- Cambiar la configuración del modo de documento.



Más Info



## RECURSO MULTIMEDIA



### 3.2.3. Utilidades para CSS

#### Inspección de las reglas CSS

En la pestaña CSS puede ver la interacción que hay entre las distintas hojas de estilo. Esta pestaña puede ser muy útil si quiere depurar una web que maneja varias hojas de estilo a la vez. Para poder pasar a visualizar de una hoja de estilo a otra, debe utilizar el selector de hojas de estilo. Al seleccionar una hoja de estilo, aparecerán, en el lado izquierdo, las propiedades u reglas de estilo asociadas a estas. Si pulsa el botón de selector de hojas de estilo, mostrará la primera hoja de estilo, por defecto. Por último, si la web tiene más hojas de estilo, puede pulsar la lista desplegable para seleccionar otra hoja de estilo.

#### Opciones del menú en la pestaña CSS

Al pulsar el botón derecho del ratón en la pestaña CSS, aparecerá un menú contextual en el que se ofrece más opciones que en la pestaña HTML. Estas opciones son contextuales y, por lo tanto, dependerán de dónde pulse dentro de la pestaña CSS. Vea a continuación una tabla con las opciones que puede encontrar:

Elemento de menú	Descripción
Agregar atributo	Se agrega un atributo o una etiqueta al elemento.
Agregar regla	Se agrega una declaración, un estilo o un selector.
Agregar regla después	Se agrega una regla o un selector después de la regla actual.
Agregar regla antes	Se agrega una regla o un selector antes de la regla actual.
Eliminar atributo	Se elimina el atributo elegido.
Eliminar regla	Se elimina la regla elegida.

#### Cambiar valores numéricos en pestaña CSS

Como ya vio anteriormente, puede cambiar los valores de las propiedades de CSS al igual que cualquiera de las otras propiedades de las herramientas F12. Para cambiar una propiedad, tan solo, debe pulsar en el valor de la propiedad y escribir el nuevo valor. Otra opción para cambiar estos

valores es utilizar las fechas de arriba y abajo, que encuentra en la pestaña CSS, para cambiar los valores.

### **Guardar cambios y buscar**

Al igual que como puede hacer con las demás pestañas, en la pestaña CSS puede también buscar propiedades, atributos, etiquetas o valores específicos. Para ello, debe escribir lo que está buscando y pulsar el botón Buscar. Al pulsar este botón, quedarán resaltadas todas las instancias que contengan la palabra que ha introducido para buscar. Si solo hay una sola coincidencia, la ventana se moverá y trasladará hasta esa coincidencia. En el caso que haya más de una coincidencia, puede desplazarse hacia adelante o hacia atrás con los botones Siguiente y Anterior.

Los cambios que realiza como, por ejemplo, agregar un atributo, no se conservarán. Por lo que, si recarga la web o navega por otra web, los cambios que haya realizado se perderán. Para guardar los cambios que haya realizado, pulsa el botón Guardar que encuentra en el lado izquierdo. Como anteriormente ha dicho, los cambios se perderán, por lo tanto, los cambios que ha guardado, se guardarán en un archivo HTML o CSS de manera local en su equipo.

### **Errores más comunes de CSS y cómo solucionarlos**

El lenguaje CSS es un lenguaje complicado de depurar. Es por ello que es difícil saber los errores que ha cometido al crear los estilos y cómo darle solución a los mismos. Sin embargo, puede echar mano de algunos consejos y buena praxis para minimizar el tiempo que invierte en la búsqueda de los errores que comete cuando utiliza el lenguaje CSS para crear sus estilos. A continuación, verá los errores más comunes:

#### **Uso innecesario del cero**

En CSS no necesita especificar la unidad si esta tiene valor cero. Por ejemplo:

Padding: 0px 0px 6px 0px;

En vez de esto, utilizaría los siguientes:

Padding: 0 0 6px 0;

Lo mismo sucede al utilizar otros elementos como:

Margin:0;

Por lo tanto, no debe dar mal uso a las unidades como px, pt, em, etc. Cuando el valor es cero. Lo único que tendría sentido para poner estas unidades es que quiera cambiar el valor más adelante.

Sin embargo, al utilizar line-height, aunque tenga un valor distinto de cero no tendrá porqué poner la unidad. Por lo tanto, quedaría como en el ejemplo siguiente:

Line-height: 3;

De todas formas, si quiere utilizar la unidad, puede hacerlo sin problema.

### **Los colores en hexadecimal siempre tendrán almohadilla**

Error: FA7D58

Correcto: #FA7D58

Correcto: color:rgb (250,125,88)

### **Código de color con valores duplicados**

No debe escribir los códigos de colores de la siguiente manera:

Color: #ffffff;

Background-color: #000000;

Border: 3px solid #ee66aa;

Por lo tanto, debe omitir los valores duplicados y escribirlos de la siguiente manera:

Color: #fff;

Background-color: #000;

Border: 1px solid #e6a;

### **Evitar repeticiones innecesarias de código**

Cuando esté programando debe evitar escribir más líneas si puede resumir el código en una sola. Por ejemplo, si está programando los estilos en CSS a veces tendrá que establecer los bordes uno a uno, pero no siempre debe ser así. Vea un ejemplo:

Border-top: 1px solid #00f;

Border-bottom: 1px solid #00f;

Border-right: 1px solid #00f;

Border-left: 1px solid #00f;

Estas cuatro líneas podría resumirlas en:

Border: 1px solid #00f;

### Duplicación necesaria con estilos en cascada

Cuando está programando estilos en cascada, puede repetir el mismo código solo si es necesario hacerlo. Por ejemplo, imagine que tiene un borde donde el único borde diferente es el de la derecha. Para realizar este código, no será necesario escribir en código cada uno de los bordes, tan solo, pondrá el borde derecho y el resto de bordes en una línea. Vea el ejemplo:

Border: 1px solid #00f;

Border-right: 1px solid #f00;

Como puede observar en este ejemplo, primero ha establecido el color de los bordes en general, y posteriormente, el color del borde diferente a los demás, que en este caso es el borde derecho. Así, ha resumido cuatro líneas de código en tan solo dos. Si lo hubiera hecho con una mala praxis hubiera quedado de la siguiente manera:

Border-top: 1px solid #00f;

Border-bottom: 1px solid #00f;

Border-right: 1px solid #f00;

Border-left: 1px solid #00f;

El resumir líneas de código, además de ser más rápido y eficiente a la hora de programar, también hace que la página cargue más rápido, ya que, el navegador tendrá que ejecutar menos líneas de código.

### Agrupación de estilos idénticos

Si diferentes partes de una web va a tener los mismos estilos, lo conveniente es agruparlos y no estar codificando una y otra vez los mismos estilos. Vea el ejemplo:

H1, p, #footer{

Font-family: Arial, Helvetica, sans-serif;

}

#### 3.2.4. Utilidades para DOM

En la Suite de Mozilla y Mozilla Firefox tiene incluida una herramienta de desarrollo llamada inspector DOM. Esta herramienta permite inspeccionar el árbol del Modelo de Objeto de Documento de los documentos basados en HTML y XML.

---

Con esta herramienta puede seleccionar un nodo DOM desde la propia estructura, o pulsando en el propio explorador. Aparte del nodo DOM, tiene otro tipo de vistas disponibles en las que se incluyen XBL binding, las reglas de estilo CSS, box model, JavaScript y estilos computados. Si quiere seleccionar objetos JavaScript u hojas de estilo de los documentos, puede hacerlo desde el propio árbol. Cuando el elemento esté seleccionado, se iluminará con un borde rojo que parpadeará, esto hace más fácil depurar el código CSS. Con esta herramienta, además de inspeccionar, también puede editar el código.

Cuando habla del DOM está hablando de una interfaz de programación de aplicación, o lo que es lo mismo, una API. Esta herramienta se utiliza para trabajar con documentos estructurados, es decir, es el modo con el que acceda y manipula el contenido de los archivos HTML y XHTML. Al ser el DOM una herramienta universal, puede acceder a él desde diferentes lenguajes como Java, Perl, PHP, Ruby, Python y C++. En cambio, en este caso va a centrar en cómo interactúa JavaScript con el DOM. Además, también centra en trabajar con documentos HTML, aunque podría emplear las mismas técnicas para documentos XML.

A finales de los noventa, los dos navegadores más potentes eran Netscape y Microsoft. Estos dos navegadores ofrecieron la posibilidad a los desarrolladores web de manipular una página web, es decir, los desarrolladores web pudieron crear el HTML dinámico, o DHTML, que conoce hoy día. Por lo tanto, el DHTML es una combinación entre CSS para los estilos, JavaScript para la funcionalidad y HTML para la estructura.

Para ambas compañías era fundamental el uso de DHTML y desarrollaron diferentes maneras de codificar el mismo comportamiento y poder acceder a las mismas partes de un documento. Así es como muchos desarrolladores siguieron creando código para que funcionase de dos maneras diferentes. De este modo, podía proveer al navegador con el código que solo él entendía. Por ejemplo, el navegador Netscape podía interactuar con sus elementos mediante el uso de una id única. Vea el ejemplo:

```
Document.layers ['myLayer'];
```

Por otro lado, Microsoft ofrecía el mismo acceso, pero realizándolo de diferente manera. Vea el ejemplo:

```
Document.all ['myLayer'];
```

Por lo tanto, programar en DHTML ocupaba mucho tiempo tanto en el desarrollo, como en el mantenimiento del mismo. Por ejemplo, los scripts que podían resumirse en pocas líneas, ocupaban cientos de líneas de código. Esto hacía que programar en DHTML fuese frustrante e ineficiente.

El W3C publicó en octubre de 1998 el DOM nivel 1. Esto sucedió gracias al esfuerzo de Netscape, Microsoft y otros miembros de la W3C lo que ofreció avanzar con el DHTML para poder aportar interactividad a las páginas web mediante el uso de JavaScript. Gracias al desarrollo del DOM se proporcionó una forma de manipular cualquier documento estructurado usando, para ello, cualquier lenguaje de programación. Es por ello que, el W3C describe el DOM como una interfaz neutral de

lenguaje y plataforma que capacita a los programas y secuencias para acceder de forma dinámica con el propósito de actualizar la estructura, el contenido y el estilo de los documentos web.

Finalmente, el DOM se estableció en Internet Explorer 5 y Netscape 6 gracias al Proyecto de estándares Web. Pero, durante algún tiempo más los desarrolladores siguieron sin hacer uso de DHTML por culpa de la mala fama que aún tenía. Sin embargo, el interés por el DOM resurgió en el año 2003 lo que llevó consigo un gran cambio para el desarrollo web. Se desterró el término DHTML y, en su lugar, se impuso el de programación DOM. La razón de este cambio de nombre se realizó por el hecho de poder eliminar la mala fama que tenía en ese momento el DHTML y que los desarrolladores pudieran confiar en la manipulación del DOM. Es por ello que actualmente, el 95% de los navegadores soportan, al menos, DOM nivel 1 y otros muchos soportan el estándar DOM nivel 2.

Gracias a la creación el DOM se ha podido ver un gran avance en cuanto el desarrollo de páginas web dinámicas y de aplicaciones web complejas. Esto hace que DOM sea una de las innovaciones más importantes en el campo del desarrollo web. Esto se debe a que los desarrolladores pueden acceder y manipular de manera sencilla las páginas XHTML como si fuesen páginas XML. De hecho, el DOM se pensó originalmente para ser utilizado en archivos XML. Definitivamente, sin importar los orígenes de esta herramienta, hoy por hoy, es una de las herramientas más esenciales que se utilizan en el desarrollo web, ya que, puede utilizarlo con una gran diversidad de lenguajes donde la única diferencia que hay es la manera de implementarlo.

### 3.3. Verificación de la compatibilidad de scripts

#### Scripts necesarios para la compatibilidad de navegadores antiguos

A partir de que se implantaron los nuevos estándares de CSS3 y HTML5, comenzaron a surgir nuevos problemas para los desarrolladores debido a la incompatibilidad de los nuevos proyectos en desarrollo o de las actualizaciones de las páginas que ya estaban en funcionamiento. Esto ocurría porque aún había navegadores que no soportaban los nuevos estándares como, por ejemplo, el navegador Internet Explorer 8.

Es por ello que no merece que preste esencial atención y esfuerzo en que las páginas web se muestren en los nuevos navegadores de la misma manera en que lo hacían en los navegadores antiguos, puesto que, los usuarios van actualizándose poco a poco. Por esa razón es mucho mejor centrarse en que los elementos vayan evolucionando conforme a la evolución de los navegadores y, por ello, los elementos vayan degradándose. A esto se le denomina progressive enhancement y puede encontrar mucha información de ello en la web para aprender qué es y cómo utilizarla.

A continuación, va a ver algunos scripts que permiten hacer compatibles sus webs en navegadores antiguos:

- **Respond.js:** Al crear una web de tipo responsive design, debe tener en cuenta el uso de las media queries que brindan compatibilidad y flexibilidad con la mayoría de dispositivos y pantallas.

Este script permite que sus media queries se ejecuten adecuadamente en los navegadores que son incapaces de soportarlas de manera natural. Hoy por hoy, solo puede soportar las medias queries max-width y min-width. Soportar estas queries puede ser útil en muchos casos, sin embargo, en otros necesite algo más. Es por ello que existen otros scripts más completos que son capaces de soportar más tipos de media queries, sin embargo, respond.js destaca por ser un script muy ligero y que, en la mayoría de los casos, será suficiente. En caso de que necesite ampliar el soporte, puede utilizar este [plug-in](#) que está recomendado por la propia página de Respond.js.

Por último, para utilizar este script, tan solo, necesite incluirlo al final de todos sus scripts.

- **History.js:** Este script permite utilizar la History API de HTML5 en navegadores que no lo tienen implementado y, de ese modo, permitir el uso de url amigables. Además, facilitará el desarrollo de aplicaciones Ajax que sean compatibles con la mayoría de navegadores existentes. Debe tener en cuenta que existe una versión de este script para navegadores HTML5 y otra para HTML4 basados en el uso de hash.
- **Modernizr:** Esta herramienta ya está muy implementada. De hecho, en muchos proyectos webs de Visual Studio, ya viene implementada. Esta herramienta permite detectar las características de HTML5 y CSS3 en el propio navegador que abre la web. Además, también permite hacer modificaciones en la misma.

En definitiva, existen un gran número de *plug-ins* que os permiten solucionar la incompatibilidad de una característica en un navegador. Si quiere comprobar que los scripts vistos anteriormente funcionan para HTML5 y CSS3, tan solo debe aplicar estas líneas de código:

```
Modernizr.load({
```

```
 Test: Modernizr.history,
```

```
 Nope: 'History.js'
```

```
});
```

Al utilizar esta instrucción, está comprobando si existe la característica history API. Si por casualidad no existiera, se cargará el script History.js. En resumen, el uso de este tipo de scripts facilita la tarea de compatibilizar y testear el funcionamiento de las aplicaciones web en diversos navegadores.

A continuación, va a pasar a analizar los problemas que puede encontrar en los diferentes navegadores. Sin embargo, antes debe saber la razón de porqué existen estos fallos, y es que, aunque existan estándares que regulen la programación web, a veces estos no se siguen. Esto sucede porque, a veces, las recomendaciones del W3C no son lo suficientemente claras y, por lo tanto, no son una guía perfecta para implementar los diferentes estándares. A veces los propios desarrolladores se ven en la encrucijada de tener que interpretar las especificaciones que aporta el W3C y, otras veces, necesitan alejarse de las especificaciones para hacer más fácil su tarea de desarrollar webs.

Además, debe entender que no existe el navegador perfecto e ideal. Es por ello que debe conocer de antemano los problemas de cada navegador y aprender a cómo solventarlos.

### **Netscape Navigator 4.x**

Hoy en día este navegador está casi fuera del mercado, ya que, su presencia en los usuarios es menor al 0,3%.

Esto significa que el soporte que tiene este navegador para CSS es demasiado básico y, aunque hay diseñadores que se dedican a diseñar para este navegador, no es rentable si compara el porcentaje de tráfico que aporta este navegador. Es por ello que la mayoría de desarrolladores solo aportan los estilos más básicos y rudimentarios, ya que, prefieren que los elementos se degraden de una manera digna en este navegador. Por lo tanto, si quiere que un diseño de degrade de manera digan para este navegador, debe crear una hoja de estilo sencilla y ocultar todos los elementos que el navegador sea incapaz de interpretar. Para ello, puede conseguirlo de dos maneras. La primera es el uso de la regla @import, la cual Netscape es incapaz de comprender. Vea el ejemplo:

```
<link rel="stylesheet" type="text/css" href="estiloBasico.css"/>
```

```
<style type_ "text/css">@import(estiloAvanzado.css)</style>
```

Como puede observar en este ejemplo, el navegador interpreta la primera hoja de estilo porque entiendo cómo vincular la hoja de estilo. Sin embargo, ignora la segunda hoja de estilo debido a que no es capaz de interpretar la instrucción @import. De este modo es como se degradaban los estilos durante muchos años. En cambio, esto conlleva a lanzar un error llamado FOUC, o lo que es lo mismo, Flash of Unstyled Content. Esto ocurre debido a que el documento no contiene ningún elemento script o link.

Sin embargo, actualmente existe otro método que aprovecha el hecho de que Netscape solo entiende el tipo de medios screen. Entonces, el truco está en que este método añadirá otros tipos de medios a la etiqueta link y, de este modo, evitará que el navegador no pueda interpretar la hoja de estilo que no convenga que interprete. Vea el ejemplo:

```
<link rel="stylesheet" type="text/css" media="screen, projection" href="estiloAvanzado.css"/>
```

### **Internet Explorer 5.x versión Windows**

Esta versión de Internet Explorer fue la que originó los hacks CSS. Es por ello que esta versión ha sido muy problemática para los diseñadores CSS, puesto que, tenía un pésimo diseño de cajas debido, precisamente, al hack.

Para explicar el modelo de cajas del W3C se entiende como algo aditivo. Por ejemplo, la anchura total de una caja sería la suma de su padding-left, border-left-width, width, border-right-width y padding right. Sin embargo, el modelo de cajas que se implementó en esta versión de Internet

Explorer es más bien sustractivo. Es por ello que la anchura general de un elemento que sería la suma del width, padding y border, se sustraen del valor global. Vea un ejemplo:

```
Div{
 Border: 5px;
 Margin: 20px;
 Padding: 20px;
 Width: 200px;
}
```

En este ejemplo puede ver cómo esto puede ser un problema, sobre todo en diseños donde se vean implicadas columnas. Por lo que en los navegadores que sí renderizan conforme al estándar la caja tendrá una anchura de 250px, o lo que es lo mismo:

5px+20px+200px+20px+5px

Finalmente, el modelo de cajas mantendría su anchura en 200px, pero la anchura general de la caja se habría reducido en 15px. Es por eso que para que Internet Explorer no dé ningún problema, debe utilizar el hack del modelo de cajas. Este elemento utiliza la propiedad voice-family para que el navegador interprete que el bloque de declaración se ha cerrado. Sin embargo, este hack daba un pequeño error en la versión del navegador Opera que había disponible en ese momento. En cambio, actualmente este problema se ha solventado en las versiones más recientes del navegador Opera.

### Resetear estilos de CSS

Aunque no utilice ningún estilo en su web, los elementos tendrán aun así un aspecto determinado. Por ejemplo, los títulos tendrán un tamaño acorde a la importancia del mismo, es decir, si es h1 tendrá un tamaño mayor que si es h6, también la web tendrá unos márgenes determinados, las citas llevarán sangría, etc. La mayoría de valores por defecto se verán de la misma forma en algunos navegadores, sin embargo, en otros no se verán igual. Por lo tanto, los navegadores en los que no se vean sus elementos igual, son los que van a dar problemas, aunque le aplique estilos a su web, ya que, el valor predeterminado seguirá estando activo. Es por ello, que debe resetear su hoja de estilo para prevenir este problema de incompatibilidad en algunos navegadores, puesto que, todos los elementos HTML ya vienen con atributos CSS implementado. Por esta razón, los espacios, márgenes, tamaños de fuente en los títulos y demás, pueden mostrarse diferente dependiendo del navegador que lo interprete, ya que, este aplicará diferentes propiedades por defecto y hará que, aunque haya establecido un estilo, los elementos de su web se muestren distintos.

Sin embargo, si resetea el código CSS va a poner los valores por defecto a cero y, por consiguiente, se aplicará el estilo de manera limpia dándole a cada elemento el estilo exacto que desee. Para poder

resetear una hoja de estilo, lo que hará es escribir el siguiente código al principio de la hoja de estilo y al final de lo que vaya programando. El código que va a ver, a continuación, se denomina código de Eric Meyer y es la manera más conocida de resetear código CSS en una web. Vea el siguiente ejemplo:

```
html, body, div, span, applet, object, iframe,
h1, h2, h3, h4, h5, h6, p, blockquote, pre,
a, abbr, acronym, address, big, cite, code,
del, dfn, em, font, img, ins, kbd, q, s, samp,
small, strike, strong, sub, sup, tt, var,
dl, dt, dd, ol, ul, li,
fieldset, form, label, legend,
table, caption, tbody, tfoot, thead, tr, th, td {
 margin: 0;
 padding: 0;
 border: 0;
 outline: 0;
 font-weight: inherit;
 font-style: inherit;
 font-size: 100%;
 font-family: inherit;
 vertical-align: baseline;
}
/* remember to define focus styles! */
:focus {
 outline: 0;
```

```
}

body {
 line-height: 1;
 color: black;
 background: white;
}

ol, ul {
 list-style: none;
}

/* tables still need 'cellspacing="0"' in the markup */

table {
 border-collapse: separate;
 border-spacing: 0;
}

caption, th, td {
 text-align: left;
 font-weight: normal;
}

blockquote:before, blockquote:after,
q:before, q:after {
 content: "";
}

blockquote, q {
```

```
quotes: "" "";
}

Por último, va a ver una manera más sencilla de poner a cero los valores por defecto que más suelen
inducir a problemas de compatibilidad. Aunque esta manera no es tan completa como la anterior,
solo anulará los elementos margin y el padding. Para ello, debe escribir la siguiente línea de código al
principio de la hoja de estilo y después de las reglas CSS que vaya creando. Vea el ejemplo:
```

```
*{margin: 0; padding: 0}
```

### **3.3.1. Parámetros para distintos navegadores**

Como ya sabe, no todos los navegadores son capaces de interpretar el código CSS, por lo que, la página podría perder sus estilos al ser mostrada. Es por ello que tiene que estar seguros de cómo se va a degradar el aspecto de la misma sin que afecte de manera sustancial al funcionamiento. Por eso, lo más importante es que el contenido esté bien estructurado. Por esta razón, aunque utilice CSS para estructurar la información en una página, debe pensar que en el documento HTML debe dar una estructura lógica a las diferentes secciones, ya que, así si topa con un navegador que no pueda interpretar el código CSS, el usuario podrá utilizar su web de forma correcta. Por lo tanto, la mejor manera que tiene para comprobar cómo se mostraría su web sin estilos es, precisamente, eliminando esos estilos. Para ello, lo único que tiene que hacer es eliminar los elementos link que contienen los vínculos con las hojas de estilos. Va a imaginar que tiene una página que contiene un título y un logotipo con un vínculo. Más abajo, tendrá el texto de la página y posteriormente el pie de página. Esta distribución, por ejemplo, sería bastante lógica.

Que su página funcione sin estilos es muy interesante para aquellos usuarios que visiten su web mediante dispositivos que tengan poca potencia de proceso y, por lo tanto, poca memoria. Hoy por hoy los dispositivos más comunes para visitar webs son los móviles y debe tener en cuenta que muchos usuarios utilizarán móviles de gama baja y ordenadores antiguos para acceder a su web. Por esta razón, estos dispositivos no interpretan la hoja CSS, ya que, no tienen la potencia suficiente para ello. Lo que hacen es aplicar una serie de atributos por defecto a los elementos como, por ejemplo, los párrafos, los diferentes tipos de títulos, enlaces, etc.

Puede encontrar casos en los que los navegadores que utilizan estos dispositivos se ven obligados a prescindir del uso de las hojas de estilo, ya que, estos navegadores son incapaces de mostrar y procesar las imágenes o, incluso, algunos de ellos no tienen capacidad para interpretar los colores. Además, debe tener en cuenta que en los sistemas UNIX puede encontrar gran cantidad de usuarios que acceden a Internet a través de terminales que solo pueden interpretar texto. Por lo que, si su

página está bien diseñada, incluso este tipo de usuarios, podrán visitarla sin ningún problema. Para poder comprobar si este tipo de usuarios pueden ver su página de forma correcta, tan solo, debe descargar Lynx e instalarlo en su equipo y, posteriormente, comprobar sus archivos webs. Esta herramienta es totalmente gratuita y funciona en los Sistemas Operativos Linux, Windows, OS/2, DOS y MAC.

En los documentos HTML, además de incluir las hojas de estilo y la información a mostrar, se debe incluir la información sobre el mismo documento. A esta información se le denomina meta información o metadata. Este tipo de información se pone en algunos elementos del documento como, por ejemplo, el elemento Title, el cual cada navegador interpreta y utiliza de una manera diferente. Otro elemento importantísimo que debe llevar su documento es el elemento meta que, hasta ahora, solo ha utilizado para indicar el tipo de codificación del documento. Vea a continuación las dos maneras más comunes de añadir el elemento meta:

```
<meta name="Propiedad" content="Valor"/>
```

```
<meta http-equiv="Propiedad" content="Valor"/>
```

Como puede observar, en la primera línea de código puede ver que aparece el atributo name. Este atributo se utiliza para indicar la propiedad a la que le asigna un valor. Por lo tanto, esta primera línea se utilizará para establecer, por ejemplo, el nombre del autor del documento, la herramienta que ha utilizado para crear el documento o la fecha en la que se ha modificado el documento. Estos datos son datos que los navegadores pueden interpretar y utilizar dependiendo de lo que les convenga. Por ejemplo, el navegador Mozilla tiene una ventana de información donde muestra el contenido de los elementos Meta.

Por otro lado, la segunda línea de código aparece, en vez de el elemento name, el elemento http-equiv. Este elemento se utilizar cuando quiere añadir algún tipo de información en la cabecera de HTTP, devolviendo ese dato de manera que parezca que el servidor web ha sido el encargado de enviar dicha información en la cabecera del archivo. Utilizar este método permite indicar de manera explícita una codificación de caracteres específica para cada página, sin tener que alterar la propia configuración de su servidor.

Los datos más que puede utilizar con más frecuencia en un documento son el nombre de la herramienta utilizada para su desarrollo, el nombre del autor y una descripción del contenido de la página. Por lo que, para indicar todos estos datos va a ver, a continuación, una lista con las tres propiedades que se utilizan para añadir tal información a su archivo:

- **Description:** Esta propiedad la utiliza para añadir una descripción de la página.
- **Generator:** Esta propiedad se utiliza para indicar el nombre del editor o herramienta usada para crear el propio documento.
- **Autor:** Aquí se indica el nombre de la empresa o persona que ha creado el documento.

A continuación, va a ver tres líneas de código que se ponen dentro del elemento HEAD y que albergan estas tres propiedades que ha visto anteriormente:

```
<meta name="author" content="Nombre de autor"/>
<meta name="generator" content="Nombre de la herramienta"/>
<meta name="description" content="Descripción de la web"/>
```

Si al crear su web utiliza una herramienta específica para programar HTML o, una herramienta más compleja de programación, lo normal es que la propia herramienta añada esas meta-etiquetas de manera automática en el HEAD de su documento HTML.

Es cierto que los diferentes buscadores poseen robots que se utilizan para examinar el contenido de las páginas y extraer información del título de las mismas, de los diferentes encabezados y de los párrafos. Sin embargo, puede manipular el funcionamiento de estos robots implementando algunas propiedades del elemento Meta. A continuación, verá cuáles:

- **Robots:** Hay algunos tipos de navegadores que tienen en cuenta este valor para procesar la página de diferente manera.
- **Description:** Hay una diversidad de navegadores que utilizarán esta propiedad como vínculo a su página.
- **Keywords:** Esta propiedad contiene una lista de palabras claves que estén relacionadas con el contenido de la web. Estas palabras, normalmente, irán separadas por comas.

Como puede observar, en esta lista aparece de nuevo la propiedad description. Sin embargo, esta propiedad en este caso va a servir para que el navegador la utilice como descripción de la página en los resultados mostrados en el buscador. Por otro lado, utiliza la propiedad keyword de la siguiente manera:

```
<meta name="keywords" content="desarrollo, programación, web"/>
```

Como vea, las palabras clave introducidas en esta etiqueta tienen que tener sentido con lo que ofrece en su página web, ya que, si no es así podría entenderse como un sitio web spam en los que se utilizan términos muy populares para conseguir visitas, pero que el contenido mostrado en este sitio no tiene nada que ver con las palabras claves ofrecidas.

Por otro lado, tiene la propiedad de robots donde debe añadir uno o más valores, ya que, son los valores que esperan encontrar los buscadores. Vea una lista de ellos:

- **None:** Este valor equivale a añadir noindex ynofollow.
- **All:** Este valor equivale a añadir index y follow.
- **Index:** Este valor indica al buscador que tiene permiso para indexar el contenido de esta página.
- **Noindex:** Este valor indica al buscador que no tiene permiso para indexar el contenido de esta página, esto suele utilizarse, por ejemplo, en la página de Aviso legal.
- **Follow:** Este valor indica al buscador que siga los hipervínculos de esta página.

- **Nofollow:** Este valor es el contrario al anterior y no permite al buscador que siga los hipervínculos de esta página.

Hoy por hoy, los buscadores utilizan un archivo llamado robots.txt en el que se escriben las instrucciones que deben seguir los robots de Internet para manipular la página. En este archivo se escribirán los valores que ha visto anteriormente. Además, actualmente cada vez está más normalizado el uso de filtros de contenido en los diferentes navegadores. Estos filtros son muy importantes y útiles, sobre todo, para los usuarios menores de edad que, cada vez más, utilizan Internet. Para que los filtros puedan funcionar de manera correcta, los desarrolladores web tienen que indicar si el contenido de la web es apto o no para niños, ya sea porque tengan un vocabulario inadecuado, promocione el uso de alcohol o tabaco, o sea sexualmente explícita, como, por ejemplo, una tienda erótica.

Según el W3C, el estándar para la clasificación de los diversos contenidos se llama PICS, o lo que es lo mismo, Platform for Internet Content Selection. Este sistema clasifica la información mediante la devolución de una cabecera HTTP de parte del servidor web. Esto quiere decir que sus documentos interpretarían esto como un elemento Meta con la estructura que verá a continuación:

```
<meta http-equiv="ics-label"
Content="(PICS-1.1 calificaciones")/>
```

A continuación, va a ver la estructura de las definiciones con la que estarán formadas estas clasificaciones:

“url organización de la calificación”

Labels for “url de su página”

Generic true ratings (clasificaciones)

Existen diversos métodos de organizaciones de calificación donde cada uno de ellos poseen su propio sistema. Sin embargo, las dos organizaciones de calificación más importantes son ICRA y RSAC. El sistema de calificación RSAC se basa en una combinación de cuatro letras v, n, s y l las cuales referencian al contenido del lenguaje utilizado, es decir, el contenido violento, la aparición de desnudos y el contenido sexual. Cada una de estas letras va seguida de un número que puede ir desde 0 a 4. Este número indica en el 0 que no existe tal tipo de contenido, hasta el 4 que sería el nivel máximo de contenido. Por ejemplo, si encuentra el valor s0, estaría indicando que no existe ningún contenido sexual. Sin embargo, si encuentra el valor v4, estaría indicando que el contenido de esa página es totalmente de tipo violento.

Por otro lado, encuentra la calificación ICRA, la cual es mucho más específica que la que ha visto anteriormente. Esta calificación está basada en códigos de dos letras y un valor numérico para cada tipo de contenido. La primera letra indica el tipo de contenido, al igual que la organización anterior, para ello tiene las letras v, n, l, c y o para los contenidos violencia, desnudos, lenguaje, chat y otros,

respectivamente. La otra letra indicaría el nivel de contenido que hay y para ello vendría indicado con una letra de la a, para mayor contenido, hasta la z, para ausencia total de ese contenido. Por lo tanto, si compara ambas organizaciones de calificación, lo que en la RSAC sería v4, en la ICRA equivaldría a va, y lo que en la RSAC sería s0, en la ICRA equivaldría a nz.

Para obtener una lista con las calificaciones para su sitio web, tan solo, debe llenar el formulario que ofrece ICRA y marcar las opciones convenientes para su sitio. En caso de que su sitio web no tenga ningún tipo de contenido sexual, violento o de desnudos, podría calificarlo con el nivel z si utiliza ICCRA y con el nivel 0, si utiliza RSAC. Ambas opciones indican que no existe contenido de naturaleza violenta o sexual, es decir, que no hay contenidos en su web que no sean aptos para menores.

Para finalizar, debe saber que los navegadores más utilizados suelen contar con un asesor de contenido que, según las distintas calificaciones, permite indicar los niveles a los que puede acceder. Por lo tanto, la configuración de este asesor quedará en manos de los propios padres, mientras que la calificación de los contenidos de su web quedará en sus manos como desarrolladores web. De esta manera, puede evitar que los menores estén expuestos a contenidos que podrían dañar su integridad.

### **3.3.2. Creación de código alternativo para diversos navegadores**

En este apartado, verá otras aplicaciones en que puede utilizar el elemento meta. Aunque, debe saber que solo se utiliza en casos muy especiales. Estas aplicaciones son actualizar la página de forma automática y redirigir a otra.

Para poder actualizar la página de manera automática, se utiliza la propiedad *refresh* con un retardo, mientras que, si quiere redirigir a otra web, debe utilizar la misma propiedad, pero utilizando una url para ello. También, puede utilizar estos métodos de manera combinada. El valor que se establece en la propiedad *refresh* será el número de segundos que deben pasar antes de que se pueda refrescar de nuevo la página.

Además de esto, si quiere redirigir al usuario a otra página, debe insertar la propiedad url con la url de la página a la que quiera redirigir al usuario.

Imagine que su web, debido al contenido que ofrece como, por ejemplo, las acciones de bolsa, deba de actualizarse con mucha frecuencia. Para hacer esta actualización continua, necesite utilizar el siguiente elemento para que el navegador, una vez el usuario haya solicitado la página, vuelva a solicitarla cada 5 minutos:

```
<meta http-equiv="refresh" content="300"/>
```

Al añadir esta línea de código a alguna de su página web, puede comprobar como cada 300 segundos, o lo que es lo mismo, cinco minutos, su página se va a actualizar de manera automática.

Debe tener en cuenta que redirigir usuarios a otra web nada más hayan llegado a su web, solo tiene sentido en muy pocos casos como, por ejemplo, que haya cambiado su dominio y, como el antiguo ya era conocido, redirija a todos los usuarios que entren por el dominio antiguo a su nuevo dominio.

Además, debe saber que las normas del W3C en cuanto a la actualización de manera automática de la página, dicen que no debería actualizarse una web o redirigir a los usuarios sin, antes, haberles avisado. Por lo tanto, si utiliza estos métodos de manera inadecuada y sin razón alguna, puede estar infringiendo las normas que dicta el W3C. Además de las aplicaciones que ha visto para el elemento meta con el atributo http-equiv, existen muchas más aplicaciones. Sin embargo, son aplicaciones que no son importantes para el tema que está tratando en este epígrafe. Algunas de estas aplicaciones pueden ser el control del almacenamiento en caché, la fecha de validez del documento o la última fecha de modificación del documento.

Además, puede encontrar diversas aplicaciones que son reconocidas solo y únicamente por los navegadores. Estas aplicaciones permiten establecer efectos de transición en las páginas, añadir información de copyright o establecer cookies.

Para ser capaces de crear una página web de manera profesional, debe conocer el lenguaje HTML y CSS y, además, disponer de una herramienta especializada para edición de código.

Sin embargo, puede combinar la edición directa de HTML con las múltiples herramientas de diseño que puede encontrar. Estas herramientas permitirán diseñar páginas, tan solo, arrastrando y soltando los elementos. Estas herramientas son editores tipo WYSIWYG, es decir, What you See Is What You Get. Estos editores están diseñados para que pueda ir viendo en todo momento la apariencia de su página. Esto es algo que facilita mucho la tarea de diseño de una web. Además, los programas de diseño frecuentemente, reúnen una serie de funciones que permiten ir más allá de solo programar código HTML. Algunas de estas herramientas son capaces de ofrecer efectos dinámicos creados mediante CSS y JavaScript. Además, son capaces de producir contenido mediante la información extraída de una base de datos o, incluso, ofrece establecer una transferencia de archivos de manera automática desde el servidor donde estén alojados estos archivos.

Es por ello que, estas herramientas no están solamente pensadas para crear páginas simples, sino que están pensadas para gestionar sitios webs de manera completa y profesional. Existen tantas aplicaciones de diseño web, como editores para HTML. Por lo tanto, como es imposible ver todas las herramientas de diseño que ofrece Internet, va a ver las más conocidas. Por ejemplo, entre las herramientas más conocidas para este fin tiene Adobe GoLive, Microsoft FrontPage y Macromedia DreamWeaver. Vea a continuación la descripción de estas herramientas:

#### **Adobe GoLive**

Seguramente, ya conoce Adobe por herramientas como Acrobat Reader o Photoshop, que son programas de diseño gráfico profesional. Sin embargo, es posible que nunca haya oído hablar de esta herramienta que es una de las que ofrece un mejor entorno de diseño web. Puede encontrar esta herramienta en su página web oficial.

La interfaz de GoLive es muy parecida a la que tienen otros programas de la misma empresa como Photoshop, Illustrator o InDesign, por lo que, si conoce alguno de estos programas, será mucho más sencillo hacerse con la interfaz de GoLive.

Cuando abra un documento en esta herramienta, aparecerá en una ventana con varias vistas y alrededor de esta ventana, aparecerán varias ventanas flotantes que contienen diversos iconos representando, de esta manera, los múltiples elementos que puede añadir en su página web, también aparecerán las distintas secciones en las que se compone su web, los estilos CSS y una información sobre el fondo.

Al igual que en la herramienta anterior, en esta herramienta aparecerán los distintos botones, en la vista WYSIWYG, que permite alternar entre las distintas vistas, desde la vista de árbol con la estructura del documento en la que puede actuar sobre las propiedades de los elementos, hasta una vista preliminar en HTML.

Además, puede acceder al código fuente de la web. En GoLive, puede trabajar tanto con documentos HTML, documentos XHTML, XHTMLMP que son para móviles, páginas WML, documentos CSS, XML, SMIL, y documentos de *script* en PHP, Perl y JavaScript. También, puede insertar objetos de InDesign y Photoshop, puesto que, estas aplicaciones son de la misma compañía.

### **Macromedia DreamWeaver**

Esta herramienta pertenece a la misma compañía que desarrolló Flash, que es el elemento que más se utilizó para crear animaciones en las webs en el pasado. Es por ello que DreamWeaver es una de las herramientas de diseño web más asentadas y, por ello, con el paso de los años se ha ido convirtiendo en la herramienta más completa y mejor. Puede utilizar esta herramienta en Windows y Mac OS X, al igual que la herramienta anterior GoLive que también puede utilizar en ambos Sistemas Operativos.

DreamWeaver tiene una interfaz de usuario muy similar a la que encuentra en Flash, por lo que, si ya está familiarizado con la herramienta Flash, será mucho más sencillo utilizar esta herramienta. Además, DreamWeaver ofrece un editor visual de páginas y, también, un editor de código. Con esta herramienta, puede crear documentos XHTML, HTML, WML, XML, CSS y documentos *scripts* con ActionScript, JavaScript, PHP, ASP, ColdFusion, entre otros. Como puede observar, este es el editor que más opciones da y que más completo es, si lo compara con herramientas que ha visto anteriormente.

En esta herramienta, el editor WYSIWYG está ubicado en el área central, y encima de este editor es donde se encuentra la barra de botones que se utiliza para insertar los elementos. Además, se encuentra más abajo una ventana con todos los atributos del elemento que haya seleccionado y, a la derecha, diversos paneles desplegables donde se hallan los estilos CSS, las capas, los fragmentos de código prefabricados y los elementos de la aplicación.

También, tiene la posibilidad de tener abierto al mismo tiempo el editor de código. Esto sucede porque DreamWeaver es capaz de sincronizar ambos editores al mismo tiempo de tal manera que si pulsa sobre un objeto en uno de los editores, se marcará el mismo objeto en el otro editor.

Por otro lado, la barra que se ubica en la parte superior y que contiene los iconos de los objetos que puede insertar en la página, es totalmente configurable. De este modo, puede indicar que muestre una lista desplegable de categorías en el lado izquierdo, o que distribuya estas categorías en modo pestaña. Esto quiere decir que la herramienta DreamWeaver es totalmente adaptable y flexible para, así, poder acomodar la herramienta como deseé.

En esta herramienta, puede observar cómo el editor de código es capaz de diferenciar de manera sintáctica los múltiples elementos del documento y, además, como ha visto en GoLive, esta herramienta también ofrece ayuda de manera continua para editar su documento. De manera, que ofrece listas desplegables de atributos, de marcar, entre otras. También, ofrece una opción en la que se resaltarán las secciones de código donde haya errores. Por ello, el programa está continuamente analizando el código que se pone.

Por otro lado, en la parte inferior del entorno, puede encontrar una ventana la cual cambia su configuración dependiendo del elemento que seleccione en su documento. De esa manera, facilita la configuración de imágenes, capas, párrafos, etc. Por ejemplo, en lo relativo a las imágenes se encuentran opciones para establecer un texto alternativo, crear un mapa de imagen, cambiar las dimensiones de la imagen, la alineación e, incluso, si tiene instalado Fireworks, tendrá la posibilidad de optimizar y editar las imágenes con el propio programa.

Como todos los programas de diseño web profesional, DreamWeaver ofrece un editor visual para hojas de estilo, aunque es cierto que, para cada hoja de estilo, tendrá que abrir una ventana para poder acceder a los selectores y, además, tendrá que abrir otra ventana si quiere configurar el selector que haya seleccionado.

En esta herramienta, las propiedades están previstas de una lista de valores, lo que hace más sencillo introducir los datos. Sin embargo, no tendrá disponible una vista preliminar en mucho de los puntos, por lo que, se hará un poco difícil prever cómo afectarán las modificaciones que vaya realizando.

Finalmente, si crea nuevas páginas tendrá la posibilidad de escoger entre varias configuraciones predefinidas de colores y diversas distribuciones, la mayoría de estas opciones están basadas en código CSS. De esta manera, será muchísimo más fácil como desarrollador, crear documentos complejos en los que, por ejemplo, puede distribuir la información en más de una o dos columnas. Para ello, solo tendrá que escoger la plantilla que deseé y añadir en contenido a cada una de las secciones de la misma.

## RESUMEN

En este tema ha aprendido cómo debe siempre incluir una descripción textual para todos los elementos de su web que no sean texto. Esto se debe a que hay usuarios que no pueden o no quieren utilizar navegadores que soporten elementos que no sean de tipo texto. Por lo que, le está brindando accesibilidad a su web y, de ese modo, está ampliando el tráfico de la misma. Para realizar esta acción, debe echar mano de la etiqueta ALT. En caso de que utilice elementos de audio o vídeo, también, será necesario incluir una descripción auditiva de los mismos, de manera, que vaya sincronizado con la presentación de los elementos.

Por otro lado, si hace una diferenciación de la información a base de colores, debe tener en cuenta que esa información se pueda comprender en el caso de que utilice un navegador que no sea capaz de interpretar colores. Para ello, lo mejor es visualizar la página en blanco y negro eliminando todos los estilos. Si es capaz de entender la información es que está organizada de manera lógica y es correcta. Para ser capaces de organizar de manera lógica la información, puede utilizar los diferentes encabezados H1 a H6, para destacar las diferentes estructuras de información. En el caso en que quiera dividir secciones, utiliza las etiquetas DIV o SPAN.

Aunque puede utilizar tablas en su página, no es aconsejable utilizarlas para dividir las secciones de su web, a no ser que esa información tenga sentido lógico si la vea de manera lineal. En caso de que utilice una tabla como base para su contenido, no debe utilizar elementos de estructura como, por ejemplo, TH que es utilizado para añadir datos en la cabecera de una tabla. En cambio, debe utilizar los elementos específicos para darle formato de manera manual.

Si incorpora scripts o applets a su web, debe tener en cuenta de que sean accesibles mediante las normas de accesibilidad de cada lenguaje. Por ejemplo, en el caso de Java que cuenta con normas de accesibilidad propias del lenguaje. Además, debe asegurarse de que los elementos de su página tengan una interfaz de usuario, la cual pueda ser manejada sin importar el dispositivo donde se esté visualizando.

Al crear scripts, debe utilizar eventos lógicos como onfocus, onblur, en vez de físicos como onmousedown. Además, debe tener en cuenta que debe evitar utilizar aquellos elementos que se hayan declarado obsoletos según la W3C, ya que, podrían crear conflictos a la hora de ser interpretados por un navegador.

Por último, siempre debe incluir en sus documentos la meta-information del mismo. Y, para ello, utiliza la etiqueta META en la cual puede añadir el autor del documento, la herramienta con la que se creó o la fecha en la que se creó o modificó.

## GLOSARIO

**Ad-hoc:** Este término se utiliza en informática para referirse a una conexión de red inalámbrica. También, se puede utilizar para especificar que algo está en remoto.

**AJAX:** es el acrónimo de JavaScript asíncrono y XML. Este lenguaje se utiliza para crear aplicaciones asíncronas que permiten la actualización de secciones específicas sin la necesidad de tener que recargar la página.

**Applet:** Es un programa que puede incrustar en un documento HTML. Cuando el navegador descarga la página web y esta contiene un Applet. Este programita se descarga en el navegador web y comienza a ejecutarse.

**Asíncrono:** La comunicación asíncrona se utiliza en muchos contextos y es un método de comunicación de mensajes entre dos o más partes, las cuales cada parte recibe y procesa el mensaje cuando sea conveniente o posible de realizar.

**Callback:** Es un tipo de función que llama a un subprograma, es decir, otra función con la finalidad de obtener algún dato de ella.

**Cross-Browser:** Es un tipo de desarrollo web que consigue que la página se vea exactamente igual en todo tipo de navegadores.

**DHTML:** Es como se denomina al HTML dinámico y es lo que aporta flexibilidad para desarrollar páginas web no estáticas.

**Directivas SSI:** Se les denomina Server Side Includes, y son un conjunto de directivas que se incluyen en las páginas HTML, pero se evalúan en el servidor cuando este solicita la página.

**Diseño Responsive:** Es el método con el que se diseñan webs de manera en que puedan ser mostradas en cualquier dispositivo sin que el usuario tenga problemas de visualización.

**DNS:** También se les denomina Servidor de Nombres de Dominio. Estos servidores contienen una lista con todos los dominios e IP almacenados en ella. Cuando un usuario pide una web determinada mediante su url, este servidor se encarga de darle la IP donde se encuentra esa web.

**DTD:** Es un documento SGML que contiene las reglas sintácticas para un tipo de documento específico. También se le denomina Definición de Tipo de Documento.

**Elementos SVG:** También denominado como Scalable Vector Graphics, o lo que es lo mismo, Gráficos Vectoriales Escalables. Es un lenguaje de marcas creado por la W3C y se utiliza para la representación de elementos gráficos.

**Expresión regular:** es una secuencia de caracteres concreta que se utiliza para hacer coincidir combinaciones de caracteres en cadenas.

**Librería:** Una librería son un conjunto de funcionalidades compiladas y codificadas en un lenguaje de programación determinado, que facilitan la programación en dicho lenguaje mediante el uso de métodos ya desarrollados por la propia librería.

**MVC:** Es un estilo de arquitectura de software denominado Modelo Vista Controlador. Este modelo separa el desarrollo de una aplicación en tres componentes distintos. La vista que compone la información que se le envía al usuario, es decir, la interfaz de usuario. El Modelo, que contiene una presentación de los datos que maneja el sistema, es decir, la lógica. Y, el controlador que es el encargado de comunicarse entre la interfaz de usuario o Vista y la lógica del programa o Modelo.

**Plug-in:** Es un programa que sirve de complemento a otro programa más grande. Los *plug-ins* permiten añadir nuevas funcionalidades ya sea a una aplicación, a un lenguaje de programación, etc.

**Shaders:** Los shaders son programas que se ejecutan en un único módulo de la tarjeta gráfica. Estos shaders se pueden cargar desde CSS para que apliquen filtros a cualquier elemento de la página. Estos filtros ponen efectos gráficos a esos elementos.

**SQL:** Sus siglas significan Lenguaje de Consulta Estructurado, este lenguaje se utiliza para la gestión de bases de datos de carácter relacional. Gracias a la utilización de álgebra y cálculos relacionales, SQL permite recuperar información de bases de datos y modificarlas de manera sencillas.

**W3C:** Se le denomina consorcio World Wide Web. Es una comunidad internacional donde las organizaciones que son miembro de este consorcio trabajan en conjunto para desarrollar los estándares Web.

**Webmasters:** Es el profesional que se dedica al mantenimiento, desarrollo y creación de webs.

**XML:** Sus siglas provienen de eXtensible Markup Language, lo que en español sería Lenguaje de Marcas Extensible. Es un metalenguaje extensible de etiquetas que permite la organización y el etiquetado de documentos. Es decir, que XML no es un lenguaje en sí, sino un sistema que permite definir lenguajes conforme a las necesidades.