

5-7-2019

Manual para principiantes: HTML, CSS, JavaScript & JQuery

Profesor: Msc. Elsner Boanerge González Ortega.

Asignatura: Desarrollo de aplicaciones Web.



Integrantes:

LEONARDO JOSE CARDOZA OROZCO - 000017518

JOSE GABRIEL PALACIOS NAVARRO - 000019093

NELSON YUBRAN CERROS UBEDA - 000018433

Contenido

HTML	3
Partes de un documento HTML	4
Etiquetas del Head	5
Link	5
TITLE	5
Lang	6
Favicon	6
Meta	6
Texto	7
Imágenes	7
Tablas	8
Listas	8
Listas Ordenadas	9
Listas Desordenadas	9
Lista de Definiciones	10
Listas Anidadas	10
Enlaces	11
Destino del enlace	12
Títulos en los enlaces	12
Contenedores(Div)	13
Form	13
Controles (Buttons, Input)	13
CSS	14
Formas de utilizar CSS	14
Estructura de un documento CSS	16
Propiedades	17
Color	17
Background	18
Border	19
Margins y Padding	21
Width y Height	21
Fonts	22

Text	23
List	24
Display.....	25
Box-Sizing	26
Position	27
JavaScript.....	30
¿Para qué nos sirve JavaScript en el desarrollo Web?.....	30
¿Qué se va a abarcar?	31
¿Cómo agregar JavaScript?	31
Tipos de datos en JavaScript	33
Tipos numéricos:	33
Arrays	35
Estructuras condicionales	36
Estructuras repetitivas.....	38
Ciclo for.....	38
Ciclo while	39
Funciones.....	40
Arrow Functions.....	40
Clases.....	41
Seleccionando un elemento con JavaScript	42
Eventos con JavaScript.....	44
JQuery	47
¿Cómo incorporar JQuery?	47
Sintaxis	47
Selectores.....	48
Eventos con JQuery	49
Efectos con JQuery	50
Hide/Show.....	50
Fading.....	51
Sliding	51
Callback.....	52
Referencias	53

HTML

HTML es un lenguaje de marcado que se utiliza para el desarrollo de páginas de Internet. Se trata de la sigla que corresponde a HyperText Markup Language, es decir, Lenguaje de Marcas de Hipertexto, que podría ser traducido como Lenguaje de Formato de Documentos para Hipertexto.

Se trata de un formato abierto que surgió a partir de las etiquetas SGML (Standard Generalized Markup Language). Concepto traducido generalmente como «Estándar de Lenguaje de Marcado Generalizado» y que se entiende como un sistema que permite ordenar y etiquetar diversos documentos dentro de una lista. Este lenguaje es el que se utiliza para especificar los nombres de las etiquetas que se utilizarán al ordenar, no existen reglas para dicha organización, por eso se dice que es un sistema de formato abierto.

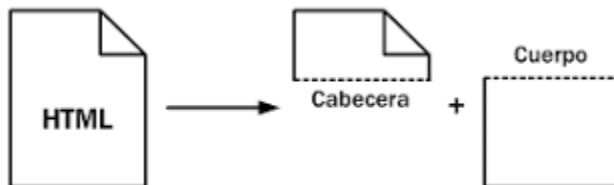
EL HTML se encarga de desarrollar una descripción sobre los contenidos que aparecen como textos y sobre su estructura, complementando dicho texto con diversos objetos (como fotografías, animaciones, etc).

Es un lenguaje muy simple y general que sirve para definir otros lenguajes que tienen que ver con el formato de los documentos. El texto en él se crea a partir de etiquetas, también llamadas tags, que permiten interconectar diversos conceptos y formatos.

Para la escritura de este lenguaje, se crean etiquetas que aparecen especificadas a través de corchetes o paréntesis angulares: < y >. Entre sus componentes, los elementos dan forma a la estructura esencial del lenguaje, ya que tienen dos propiedades (el contenido en sí mismo y sus atributos).

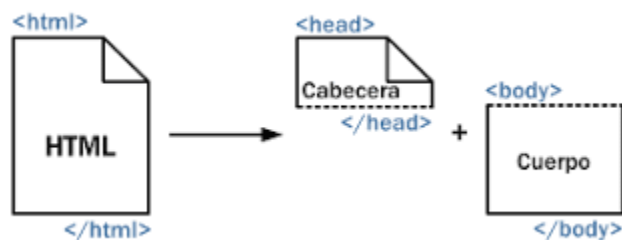
Partes de un documento HTML

Un documento HTML ha de estar delimitado por la etiqueta `<html>` y `</html>`. Dentro de este documento, podemos distinguir dos partes principales: La cabecera y el cuerpo.



El encabezado, delimitado por `<head>` y `</head>` donde colocaremos etiquetas de índole informativo como por ejemplo el título de nuestra página.

El cuerpo, por las etiquetas `<body>` y `</body>`, que será donde colocaremos nuestro texto e imágenes delimitados a su vez por otras etiquetas.



`<html>`: indica el comienzo y el final de un documento HTML. Ninguna etiqueta o contenido puede colocarse antes o después de la etiqueta `<html>`. En el interior de la etiqueta `<html>` se definen la cabecera y el cuerpo del documento HTML y todo lo que se coloque fuera de la etiqueta `<html>` se ignora.

`<head>`: delimita la parte de la cabecera del documento. La cabecera contiene información sobre el propio documento HTML, como por ejemplo su título y el idioma de la página. Los contenidos indicados en la cabecera no son visibles para el usuario, con la excepción de la etiqueta `<title>`, que se utiliza para indicar el título del documento y que los navegadores lo visualizan en la parte superior izquierda de la ventana o pestaña del navegador.

<body>: delimita el cuerpo del documento HTML. El cuerpo encierra todos los contenidos que se muestran al usuario (párrafos de texto, imágenes, tablas). En general, el <body> de un documento contiene cientos de etiquetas HTML, mientras que el <head> no contiene más que unas pocas.

```
1  <!DOCTYPE html>
2  <html lang="en"></html>
3  <head>
4  |   <title>Document</title>
5  </head>
6  <body>
7
8  </body>
9  </html>
```

Etiquetas del Head

Link

El elemento HTML <link> especifica la relación entre el documento actual y un recurso externo. Los usos posibles de este elemento incluyen la definición de un marco relacional para navegación. Este elemento es más frecuentemente usado para enlazar hojas de estilos.

```
<head>
  <title>Document</title>
  <link rel="stylesheet" href="">
</head>
```

TITLE

Un elemento del HEAD visible desde el navegador, muy importante para los buscadores pues es el texto que se visualiza en los resultados de búsqueda.

```
<head>
  <title>Document</title>
</head>
```

Lang

Se utiliza el atributo lang para establecer el idioma del documento.

```
<html lang="en">
```

Favicon

Un favicon es la pequeña imagen que se muestra en la pestaña del navegador o en la lista de marcadores (favoritos).

```
<head>
  <title>Document</title>
  <link rel="stylesheet" href="">
  <link rel="shortcut icon" href="favicon.ico" type="image/x-icon">
</head>
```

Meta

Las metaetiquetas, etiquetas meta o elementos meta (también conocidas por su nombre en inglés, metatags o meta tags) son etiquetas HTML que se incorporan en el encabezado de una página web y que resultan invisibles para un visitante normal, pero de gran utilidad para navegadores u otros programas que puedan valerse de esta información.

Su propósito es el de incluir información (metadatos) de referencia sobre la página: autor, título, fecha, palabras clave, descripción, etc.

```
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta http-equiv="X-UA-Compatible" content="ie=edge">
  <title>Document</title>
  <link rel="stylesheet" href="">
  <link rel="shortcut icon" href="favicon.ico" type="image/x-icon">
</head>
```

Texto

En un documento HTML existen dos tipos de etiquetas HTML: las etiquetas que contienen fragmentos de texto (se utilizan para dar significado a esas palabras o fragmentos) y las etiquetas que agrupan conjunto de información (fragmentos de texto y/u otras etiquetas).

La etiqueta `<p>` y `</p>` se utiliza para párrafos.

La etiqueta `` se utiliza para fragmentos de texto importante o palabras clave.

La etiqueta `` se utiliza para fragmento de texto sin significado (útil para seleccionar).

La etiqueta `<cite>` se utiliza para fragmento de texto con el título de un trabajo creativo: obras, libros...

La etiqueta `<small>` se utiliza para anotaciones menores, pequeñas puntualizaciones.

```
<p>Lorem ipsum dolor sit amet consectetur adipisicing elit. A corporis quod quis libero dolorem.  
Fugit libero asperiores at <strong>iste velit</strong> cumque veniam magni optio, labore,  
<span>quidem</span> eos vel sapiente. <cite>Aliquid.</cite>  
</p>  
  
<p>  
  Lorem, ipsum dolor sit amet consectetur adipisicing elit. Incidunt rerum deleniti quidem voluptas repudiandae aspe  
  Consectetur <small>quasi maiores error cupiditate</small> eaque libero quae suscipit vitae autem expedita.  
</p>
```

Imágenes

Para poner una imagen simple en una página web, utilizamos el elemento ``. Este es un elemento vacío (lo que significa que no contiene texto o etiqueta de cierre) que requiere de por lo menos un atributo para ser utilizado — `src` (a veces llamado completamente como, `source`). El atributo `src` contiene una ruta que apunta a la imagen que quieres poner en la página, que puede ser una URL relativa o absoluta.

```
<img src="" alt="">
```


Tablas

En documentos HTML una tabla puede ser considerada, resumidamente, como un grupo de filas donde cada una contiene a un grupo de celdas. Esto es conceptualmente distinto a un grupo de columnas que contiene a un grupo de filas, y esta diferencia tendrá un impacto en la composición y comportamiento de la tabla.

Como muchas otras estructuras de HTML, las tablas son construidas utilizando elementos. En particular, una tabla básica puede ser declarada usando tres elementos, a saber, table (el contenedor principal), tr (representando a las filas contenedoras de las celdas) y td (representando a las celdas).

```
<table>
  <tr>
    <td>Celda 1</td>
    <td>Celda 2</td>
    <td>Celda 3</td>
  </tr>
  <tr>
    <td>Celda 4</td>
    <td>Celda 5</td>
    <td>Celda 6</td>
  </tr>
</table>
```

Listas

Las listas en HTML nos permiten crear conjuntos de elementos en forma de lista dentro de una página, todos los cuales irán precedidos, generalmente, por un guion o número.

Los tipos de listas en HTML son los siguientes:

- Listas ordenadas
- Listas desordenadas
- Listas de definiciones

Listas Ordenadas

Las listas en HTML ordenadas son aquellas que nos muestran los elementos de la lista en orden. Para representar el orden tendremos los elementos numerados. Es decir, cada uno de los elementos irá precedido de un número o letra que establezca su orden.

Las listas en HTML ordenadas se representan mediante el elemento OL.

Cada uno de los elementos de la lista ordenada se representará mediante el elemento LI.

```
<ol>
  <li>Elemento1</li>
  <li>Elemento2</li>
  <li>Elemento3</li>
  <li>Elemento4</li>
</ol>
```

Listas Desordenadas

Las listas desordenadas en HTML nos sirven para mostrar los elementos sin ningún tipo de orden, simplemente precedidos por una viñeta que puede ser un punto, un cuadrado, etc.

Para definir una lista desordenada en HTML utilizamos el elemento ul.

Para representar los elementos de la lista desordenada utilizamos el mismo elemento que con las listas ordenadas, es decir, el elemento li.

```
<ul>
  <li>Elemento1</li>
  <li>Elemento2</li>
  <li>Elemento3</li>
  <li>Elemento4</li>
</ul>
```

Lista de Definiciones

Las listas en HTML de definiciones en HTML nos sirven para montar listas en las que tenemos la estructura valor y definición. Suelen ser listas para definir términos, como si fuese un diccionario, si bien pueden ser cualquier par valor-definición.

Las listas en HTML de definiciones en HTML se construyen mediante el elemento `dl` `<dl>....</dl>`

En este caso, dentro de las listas en HTML de definiciones tenemos dos elementos anidados, el que representa al valor `dt` y el que representa a la definición `dd`. De esta forma la estructura de las listas en HTML de definiciones es la siguiente:

```
<dl>  
  <dt>Término1</dt>  
  <dd>Definición 1</dd>  
  <dt>Término 2</dt>  
  <dd>Definición 2</dd>  
  
  <dt>Término N</dt>  
  <dd>Definición N</dd>  
</dl>
```

Listas Anidadas

Cuando estemos manejando listas podemos anidar unas en otras independientemente del tipo de lista que estemos anidando.

Para crear listas anidadas en HTML simplemente tenemos que hacer que el elemento de una de las listas sea a su vez una lista. Es decir, el esquema de listas sería parecido al siguiente:

```

<ul>
  <li>Elemento 1</li>
  <li>Elemento 2</li>
  <li>
    <ol>
      <li>Elemento 2.1</li>
      <li>Elemento 2.2</li>
      <li>Elemento 2.N</li>
    </ol>
  </li>
  <li>Elemento 3</li>
  <li>Elemento N</li>
</ul>

```

Enlaces

Lo más importante de los documentos HTML son los enlaces. Ya que mediante los enlaces en HTML podemos comunicar una página con otra. De esta forma, enlazando documentos HTML podemos acabar tejiendo lo que es Internet.

Para crear un enlace en HTML utilizamos el elemento A con la siguiente sintaxis:

```
<a>Contenido del enlace</a>
```

Pero solo con esto el enlace no tendrá mucha utilidad ya que el principal objetivo del enlace es enlazar a un destino. Para poder indicar el destino de un enlace utilizamos el [atributo href][Href]. En valor del [atributo href][Href] puede ser cualquier URI que represente un recurso. Es decir, una página, una parte de una página, una URL genérica, un archivo, ... De esta forma el enlace en HTML lo crearemos con la sintaxis:

```
<a href="URI">Contenido del enlace</a>
```

A modo de ejemplo podríamos tener los siguientes enlaces:

```

<a href="documento.html">Enlace a un documento</a>
<a href="documento.html#resumen">Enlace a una parte de un documento</a>
<a href="http://www.manualweb.net">Enlace a una web</a>
<a href="http://www.manualweb.net/tutorial-html/">Enlace a un directorio</a>
<a href="miimagen.png">Enlace a una imagen</a>
<a href="mimusica.mp3">Enlace a un archivo de sonido</a>

```

Destino del enlace

Pero, ¿dónde se abre el enlace? Pues por defecto y si no hemos configurado nada en el navegador web que estemos utilizando el enlace se abre en la misma ventana en la que tengamos el enlace.

Si bien, en el enlace, podemos indicar el destino que queremos darle a dicho enlace. Eso lo podemos hacer mediante el atributo target. Los posibles valores que admite el atributo target son:

- `_blank`, el agente de usuario intentará abrir el enlace en una nueva ventana. La nueva ventana no tendrá nombre.
- `_self`, el agente de usuario intentará abrir el enlace en el mismo marco donde está en código actual.
- `_parent`, el agente de usuario intentará abrir el enlace en el frameset inmediatamente superior al que se encuentra la página. Esto suele suceder cuando tenemos el enlace en un área de frames.
- `_top`, el agente de usuario intentará abrir el enlace en la ventana padre. En el caso de que exista un frameset lo eliminará y se hará con toda la ventana.
- `nombre_marco`, se podrá indicar el nombre de un frame. En este caso el agente de usuario intentará abrir el enlace en el frame que coincida con el nombre. En el caso de no existir un frame con ese nombre lo abrirá en una nueva ventana, asignándole dicho nombre.

Así podremos tener el siguiente código:

```
<a href="enlace.html" target="_blank">Abrir enlace en una nueva ventana</a>  
<a href="enlace.html" target="_top">Abrir enlace en la ventana superior</a>
```

Títulos en los enlaces

El enlace en HTML, tal y como lo hemos visto hasta ahora, sirve para enlazar contra un recurso de la web: servidor, directorio, dominio, ... Y lo que en mayor o menor medida describe lo que enlazamos es el contenido que encontramos entre las etiquetas A.

Si bien el elemento A nos ofrece un atributo llamado title. En el atributo title podemos describir de una forma textual el destino del enlace. Esto servirá no solo al usuario para que pueda obtener más información de dónde va, si no a las máquinas a la hora de asignar un nombre a la URI sobre la que estamos enlazando.

Un ejemplo sería:

```
<a href="http://www.manual.net" title="Web de Manuales y Tutoriales de Programación">Ir a Manual Web</a>
```

Contenedores(Div)

div de "division" -división . Sirve para crear secciones o agrupar contenidos.

```
<div></div>
```

Form

Form es una etiqueta de HTML que representa un formulario. En este formulario se agregan los diferentes campos de entrada de datos o de confirmación, así como los botones, que como mínimo ha de haber uno, el de envío. También pueden agregarse más botones como el de Restablecer que pone el formulario en blanco.

```
<form action=""></form>
```

Controles (Buttons, Input)

-La etiqueta de HTML <button> representa un elemento cliqueable de tipo botón que puede ser utilizado en formularios o en cualquier parte de la página que necesite un botón estándar y simple de aplicar. De forma predeterminada, los botones HTML se presentan con un estilo similar en todas las plataformas, estos estilos se pueden cambiar utilizando CSS.

```
<button></button>
```

- El elemento HTML <input> se usa para crear controles interactivos para formularios basados en la web, que reciban datos del usuario. La forma en que <input> funciona varía considerablemente dependiendo del valor de su atributo type.

```
<input type="text">
```

CSS

CSS, las siglas para *Cascading Style Sheets*, u Hojas de estilo en cascada, es una tecnología que nos permite crear una página web de manera más exacta; o, siendo más específicos, que nos permite editar una página web. Si bien se puede decir que HTML (por favor, leer antes el manual de HTML) es la estructura de una página web, CSS es la forma de darle ese atractivo visual a la página, permitiéndonos ordenar los textos, darles color, poner un fondo, crear banners, etc., en fin, todo para una edición de página web.

Formas de utilizar CSS

Si bien es recomendable utilizar un editor de texto más profesional, crear hojas de estilo con CSS (cabe recalcar que CSS es una forma de trabajar hojas de estilo, pero existen más) con el simple bloc de notas que traen todas las computadoras.

La base, y lo primordial, es crear un archivo con extensión ".css". Luego de esto, ya podemos comenzar a editar nuestra hoja de estilo. Existen tres formas para utilizar CSS:

La primera es añadir el atributo "style" en la apertura de una etiqueta, y luego definir la característica que esa etiqueta tendrá, por ejemplo:

```
1  <!DOCTYPE html>
2  <html lang="es-ES">
3  <head>
4      <meta charset="utf-8">
5  </head>
6  <body>
7      <p style=color:red>
8          La letra de este párrafo se pondrá en rojo.
9      </p>
10 </body>
11 </html>
```

Nótese primero "<p" para aperturar la etiqueta de párrafo, luego la palabra "style", seguido de la cualidad "color:red>".

La siguiente forma, es crear la hoja de estilo, pero dentro la cabecera, o etiqueta "head" del archivo html, por ejemplo:

```
1  <!DOCTYPE html>
2  <html lang="es-ES">
3  <head>
4      <meta charset="utf-8">
5      <style type="text/css">
6          p{
7              color:red;
8              background-color:blue;
9          }
10     </style>
11 </head>
12 <body>
13     <p>
14         La letra de este párrafo se pondrá en rojo y
           el fondo será azul.
15     </p>
16 </body>
17 </html>
```

Aquí ya apreciamos como se trabaja una hoja de estilo, con la única diferencia que se tiene que especificar lo que es (una hoja de estilo), en este caso, con la etiqueta “<style type=“text/css”> (recuerden cerrar esta etiqueta “/style”).

Luego, se escribe la o las (en este caso “p”) etiquetas que se verán afectadas por este estilo (si fuera las, se enlazan con una coma (,) y se abre llave ({), luego, la cualidad, o cualidades que tendrá esta (o estas) etiqueta.

La tercera y última forma, y la que utilizaremos para este manual, es crear un archivo con la extensión “.css”, y referenciarlo dentro del archivo html, así:

```
1  <!DOCTYPE html>
2  <html lang="es-ES">
3  <head>
4      <meta charset="utf-8">
5      <link rel="stylesheet" href="ejemplo.css">
6  </head>
7  <body>
8      <p>
9          La letra de este párrafo se pondrá en rojo y
           el fondo será azul.
10     </p>
11 </body>
12 </html>
```


Dentro de la etiqueta link (no se cierra esta etiqueta más que con el signo ">") definimos el tipo de relación con la etiqueta "rel", en este caso "stylesheet", u hoja de estilo en español, y con la etiqueta "href" ponemos la referencia, o sea, la hoja de estilo a utilizar. En el caso de este ejemplo se utilizó el nombre "ejemplos.css". Luego, todos los cambios hechos en la hoja de estilo serán automáticamente aplicados a este archivo html; si la hoja de estilo no estuviera en la misma carpeta que el archivo html, se debe especificar la ruta, así que es recomendable mantener estos dos archivos (y los demás, si se usan, archivos html) juntos.

Estructura de un documento CSS

El documento CSS, a diferencia del html, no necesita algo que lo especifique como tal (en el caso de html, "<!DOCTYPE html>" y la etiquetas propias <html> y </html>, simplemente el documento tiene que tener la extensión ".css" al final.

Este documento, que llamaremos ahora "hoja de estilo", cumple un orden jerárquico, las "reglas", llamadas así en CSS, más específicas se aplicarán primero, luego las generales, o sea, si tengo un cambio de color en la etiqueta "body", pero también en las etiquetas "p", este último cambio será el que se aplique.

Sabiendo esto, debe quedar claro que estas reglas son aplicables en las etiquetas que harán un cambio visual en la página web, o sea, "header", "footer", "body", "p", "pre", "div", todas las etiquetas "h" con su número posterior, por supuesto, asterisco (*) para especificar que el cambio será en todo el documento, etc.; también se puede ser mucho más específico, por ejemplo, se pueden anidar etiquetas dentro de etiquetas, por ejemplo, si quisiera aplicar un estilo a todos los párrafos dentro de las etiquetas div, sería "div p", también, los estilos son aplicables en las clases, simplemente se escribe un punto (.) seguido del nombre de la clase, igualmente, se pueden aplicar en los id's, utilizando, en vez de un punto, el numeral (#), seguido del nombre del id. Otra forma de aplicar hojas de estilo es con el selector de adyacencia, con esto, se aplicará al primer elemento que sigue al especificado de primero, o sea, si yo quisiera aplicarle un estilo a la primera etiqueta "p" que le siga a la finalización de un "div", sería así: "div + p", o sea, se aplicará al primer elemento "p" que esté al mismo nivel de los elementos "div"; entre muchas otras formas.

Estas reglas, se abren y cierran con llaves; dentro de estas se escribirán especificaciones que se cumplirán en la página al cumplirse esta regla, por ejemplo, así:

```
1 p{  
2   color:red;  
3   background-color:blue;  
4 }  
5 |
```

La estructura termina siendo: El selector, que sería la etiqueta “p”; la propiedad que sería “color” y el valor, que sería “red”, no olviden en juntar estos dos mediante dos puntos (:) y terminar cada declaración (que sería cada característica que poseerá la regla, “color:red”) con un punto y coma (;).

Propiedades

Color

En los ejemplos anteriores, se ha utilizado la propiedad “color” varias veces, esta sirve para cambiar el color de algo; sin ninguna propiedad acompañándola, “color” funcionará para cambiar el color de la letra solamente. La sintaxis es:

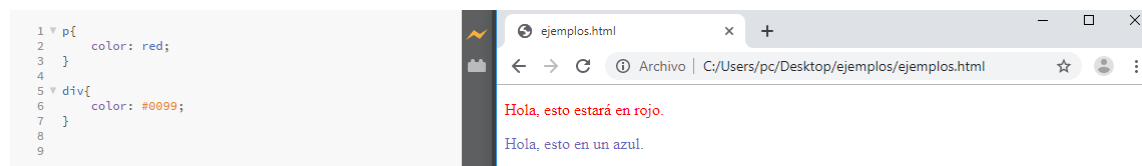
color: “color a elegir”.

Se puede utilizar el nombre (en inglés del color), o por cantidad de rojos, verdes y azules (RGB), valores HEX, etc., un ejemplo más específico sería:

color: red;

color: #0099;

Este sería el resultado:



Como se logra apreciar, ambos casos ofrecen el mismo resultado, simplemente es cuestión de la preferencia de cada usuario.

Background

Esta propiedad sirve para editar el fondo de la página; por sí sola, “background” funcionará para definirle un color de fondo (a menos que con la propiedad url se especifique que es una imagen –se verá más adelante), de la misma manera, se le puede agregar la propiedad “color” para ser más específico; estos dos ejemplos harán lo mismo:

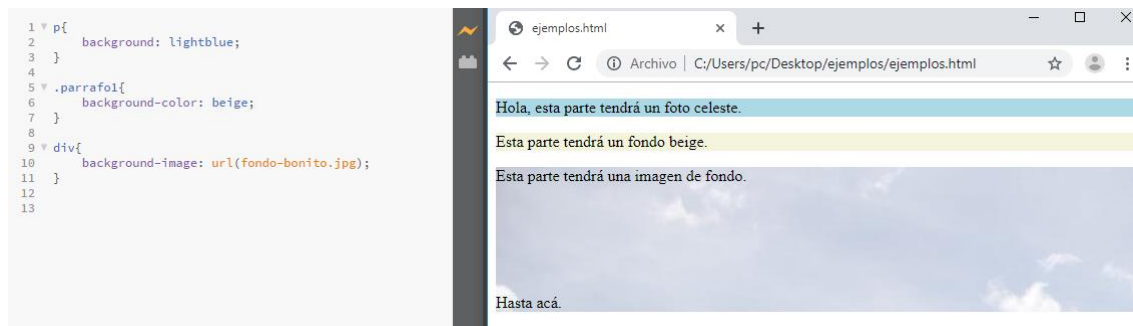
```
background: red;
```

```
background-color: red;
```

Acompañada de la propiedad “image”, se definirá una imagen de fondo, de esta manera:

```
background-image: url("Desktop/imagenes/imagenbonita.jpg");
```

Este sería el resultado:



Se utiliza “url” para especificar la ruta donde se encuentra la imagen, si la imagen se encuentra en la misma carpeta, simplemente se pondrá el nombre de la misma.

También se pueden las propiedades a esa imagen, acompañando “background” con algunas de las siguientes propiedades: “repeat”, “position” y “attachment”. Por ejemplo:

```
background-image: url("Desktop/imagenes/imagenbonita.jpg");
```

```
background-repeat: no-repeat;
```

```
background-position: 2em 1.5cm;
```

```
background-attachment: fixed;
```

Aquí se especifica la ruta de la imagen, que esta no se repetirá (o sea que sólo saldrá una imagen en el fondo), la posición de la misma (em es reemplazable por cualquier otra unidad de medida de CSS, px, %, em, rem, etc.) y que esta tendrá un comportamiento “fixed”, o sea, que no bajará con el *scroll* de la página, estará estática.

Todas estas propiedades pueden ser definidas en una misma declaración, sin importar mucho el orden de las propiedades de la imagen de esta manera:

```
background: url("Desktop/imagenes/imagenbonita.jpg") no-repeat 2em 1.5cm fixed;
```

El orden de estas no importa.

Border

La propiedad "border" sirve para crear un borde. Esta se acompaña de la propiedad "style", seguido del tipo de borde como valor. Existen los siguientes tipos: "dotted", "dashed", "solid", "double", "groove", "ridge", "inset", "outset", "none", "hidden". Por ejemplo:

```
border-style: solid;
```

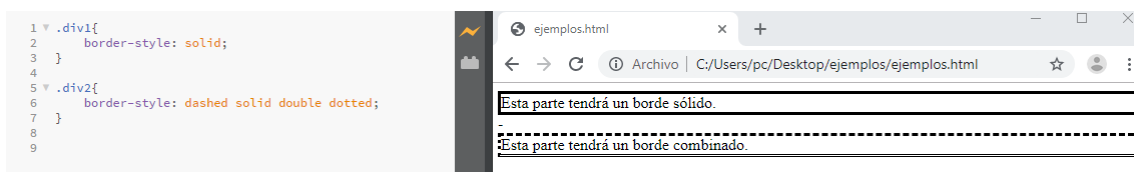
Se pueden definir cuatro tipos de borde por declaración, en el orden de arriba, derecha, abajo, izquierda, por ejemplo:

```
border-style: solid dashed double ridge;
```

También se puede acompañar de la propiedad "color" para darle un color (Si no se especifica un color, este se hereda):

```
border-color: red;
```

Tenemos un ejemplo utilizando lo anteriormente explicado:



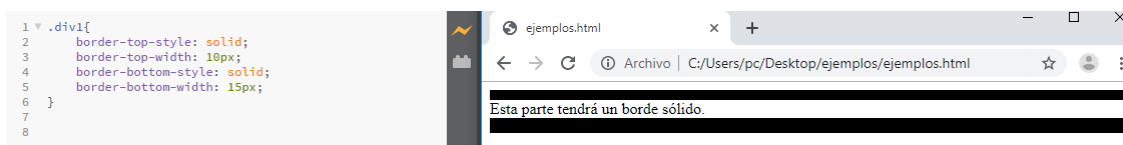
También se puede especificar la anchura del borde, con la propiedad "width", utilizando las medidas anteriormente mencionadas (px, %, em, rem, etc.), o usando uno de los tres valores pre-definidos: "thin", "medium" o "thick". Igualmente, con el color, se puede poner hasta cuatro valores de anchura, de esta manera:

```
border-width: 2px 5px 10px 1px;
```

También, se puede ser más específico, escribiendo antes de las propiedades anteriormente mencionadas, las propiedades "top", "right", "bottom" y "left". Por ejemplo:

```
border-top-style: solid;
```

Veamos este ejemplo a continuación:



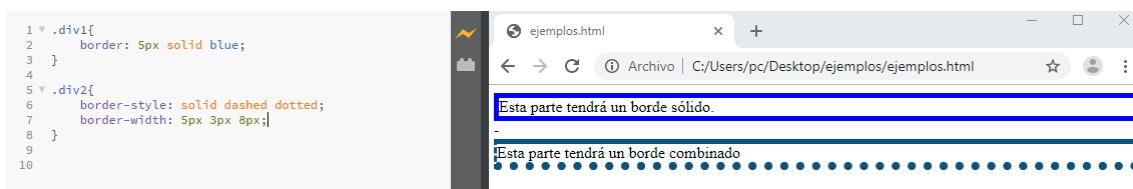
Como podemos apreciar, se especifican dos bordes, el de arriba y el de abajo, aparte de que se les da una anchura distinta.

Si se utiliza un valor, todos los bordes lo tendrán. Si se especifican dos valores, el borde de arriba y el de abajo tendrán el primer valor y el segundo valor lo tendrán los de izquierda y derecha. Si se especifican tres, el primer valor lo tendrá el borde de arriba, el segundo valor lo tendrán los bordes de izquierda y derecha, y el tercero lo tendrá el borde de abajo. Si se utilizan cuatro, será como se mencionó anteriormente.

También es posible aplicar todas las propiedades en una sola declaración:

`border-top: 5px solid red;`

Por ejemplo:

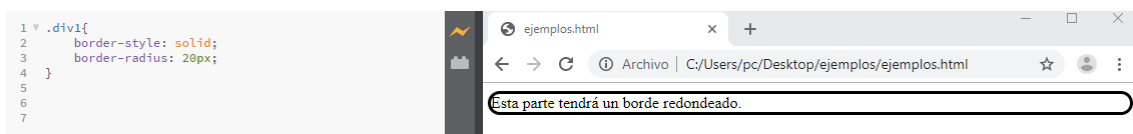


Como se aprecia, para el primer “div” se utilizó una combinación de declaraciones en una sola línea, dándole anchura, tipo de borde y color. En el siguiente div, se utilizan las combinaciones previamente mencionadas, tres tipos de borde para que se vea la jerarquía anteriormente explicada y tres tipos de anchura.

Por último, el borde puede ser redondeado, agregándole un radio al mismo, con la propiedad “radius”, por ejemplo:

`border-radius: 5px;`

Quedando de esta manera:



Dándonos como resultado un borde de forma redondeado.

Margins y Padding

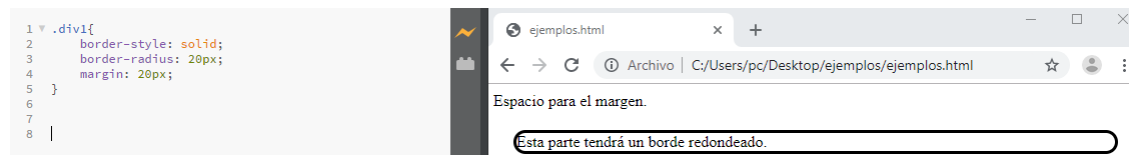
La propiedad “margin” sirve para crear distancia entre un borde definido y el espacio dado a la etiqueta. Se trabaja con unidades de medida y de la misma forma que los bordes, o sea, se pueden definir hasta cuatro, siguiendo la misma lógica anteriormente explicada, o especificando que margen será el afectado, por ejemplo:

margin: 25px;

margin-top: 25px;

margin: 25px 10px 8px 15px;

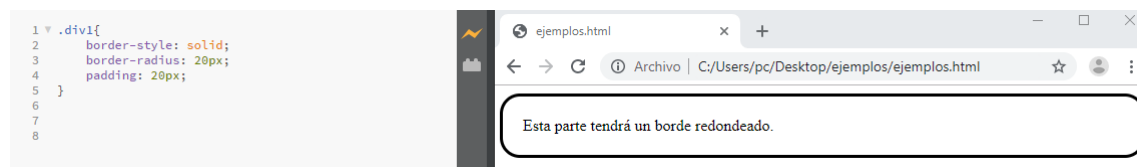
Como por ejemplo:



Como se puede apreciar, se deja un margen a todos los lados, debido a que no se especificó lugar del mismo, simplemente la existencia y el valor del margen.

También tiene valores pre definidos, “auto”, para centrar el elemento horizontalmente dentro de su contenedor. Y la propiedad “inherit” para heredar un margen.

La propiedad “padding” sirve para crear una distancia entre los bordes y el contenido, o sea, un relleno. Funciona de la misma manera que el “margin” pero sin los valores pre definidos “auto” ni “inherit”.

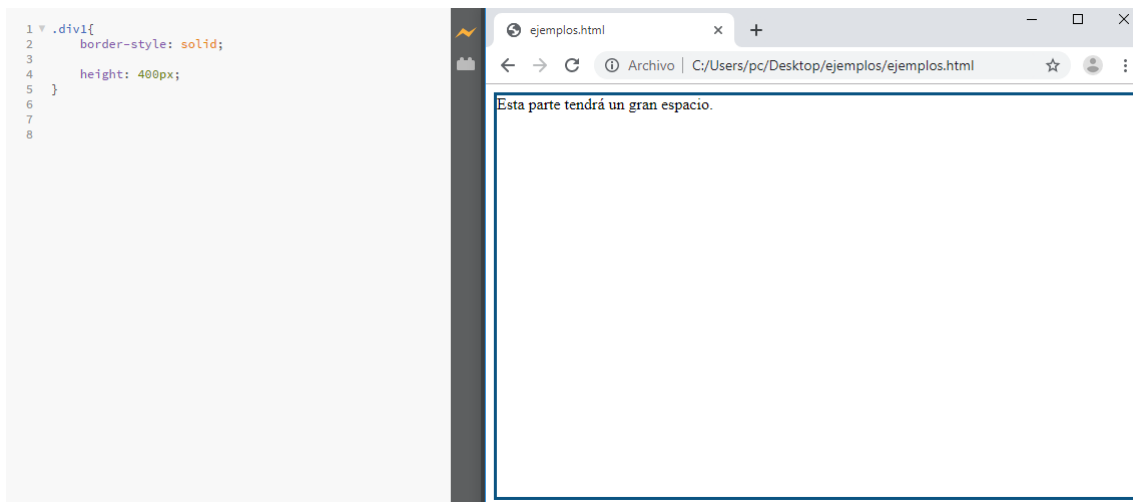


Como podemos apreciar, se deja un espacio de 20px entre el contenido y el margen.

Width y Height

La propiedad “width” se utiliza para definir la anchura de un elemento, utilizando la unidad de medida que se prefiera y valores predefinidos. La propiedad “height” sirve para definir la altura de un elemento, funcionando de la misma manera que la propiedad “width”.

Pondremos un ejemplo sólo para la propiedad “height”, debido a que se ha utilizado anteriormente la propiedad “width” en los ejemplos anteriores, resultando así:



Como podemos apreciar, el borde llega a cubrir toda la parte del divisor, el cual posee una altura especificada de 400px.

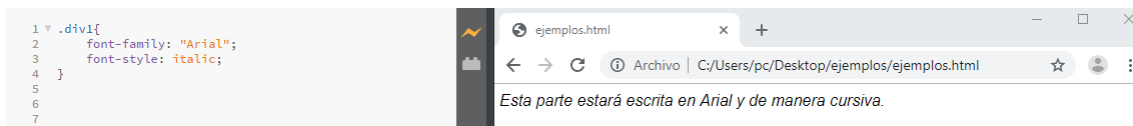
Fonts

La propiedad “font” sirve para definir todo lo que tenga que ver con las fuentes; tamaño, familia de fuente y estilo de fuente.

La propiedad “font” acompañada de la propiedad “family” sirve para definir la familia a la que la letra pertenecerá. Si se acompaña, en cambio, con la propiedad “style” se definirá que estilo tendrá esta letra, de esta manera:

font-family: “Arial”;

font-style: italic;



Como podemos apreciar, se define la letra Arial, y el estilo italic, dándonos ese resultado.

También, “font” se puede combinar con otras propiedades, como por ejemplo “size” para definirle un tamaño y “weight” para definirle el “peso”, lo que sería letras normales, o negritas

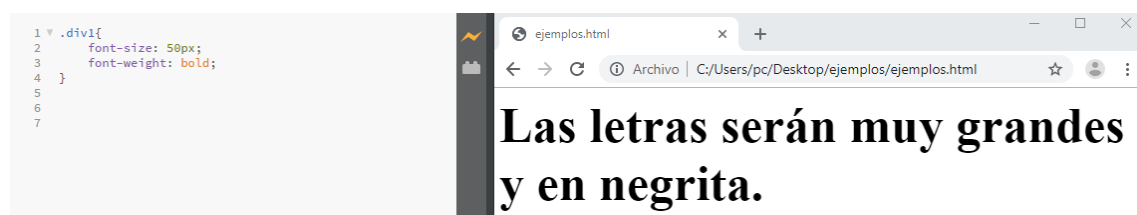
(también se puede utilizar, para esto, la etiqueta en html, recuerden). Utilicemos estas dos propiedades para el siguiente ejemplo:

```
font-size: 50px;
```

```
font-weight: bold;
```

También, se recuerda que la medición no es necesaria en pixeles, y, se da el consejo de que no es necesario especificar “bold” o “normal”, funciona con valores numéricos sugeridos, por ejemplo:

```
font-weight: 100;
```



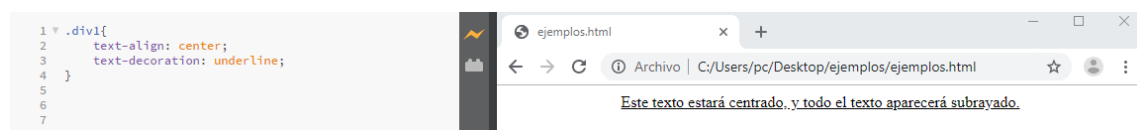
Para el ejemplo, como se aprecia, se especifica el tamaño, 50px y el peso como letras negritas, o bold.

Text

Si bien, esta propiedad ya la hemos alterado explícitamente con la propiedad sola “color”, se puede alterar el texto de una manera más específica con la propiedad “text”, que por sí sola no hace nada, pero, el texto se puede alinear si se utiliza con la propiedad “align”, también, se puede modificar la decoración con la propiedad “decoration”, por ejemplo:

```
text-align: center;
```

```
text-decoration: underline;
```



Donde se especifica que el texto tendrá una alineación centrada y el texto estará subrayado.

También, existen, para acompañar a la propiedad “text”, las siguientes propiedades: “transform” acompañado de valores como “uppercase” y “lowercase”; esto sirve para que todo el texto esté o en minúscula o en mayúscula, de la siguiente manera:

```
text-transform: uppercase;
```

También, existe la propiedad “indent”, para especificar la sangría, de esta manera:

```
text-indent: 50px;
```

Especificando un valor numérico que representará el espacio de la sangría. También existe la propiedad –menos usada que las anteriores- “spacing”, para dejar un espacio entre cada letra, como para editar un título, de la siguiente forma:

```
letter-spacing: 1px;
```

Nótese que esto se aplica a las letras, no al texto, por eso se utiliza la propiedad “letter”, no “text”. Y no, no existe la declaración “text-spacing”.

Nota: Antes de continuar con las siguientes propiedades, se insta a todos los lectores de este manual, a practicar todas las posibilidades que ofrecen estas propiedades. Si las repasáramos todas, sería un manual de 100 páginas sólo para CSS, y ese no es el caso, se recuerda que es un manual de principiantes, así que, por favor, practique las propiedades anteriormente explicadas antes de continuar.

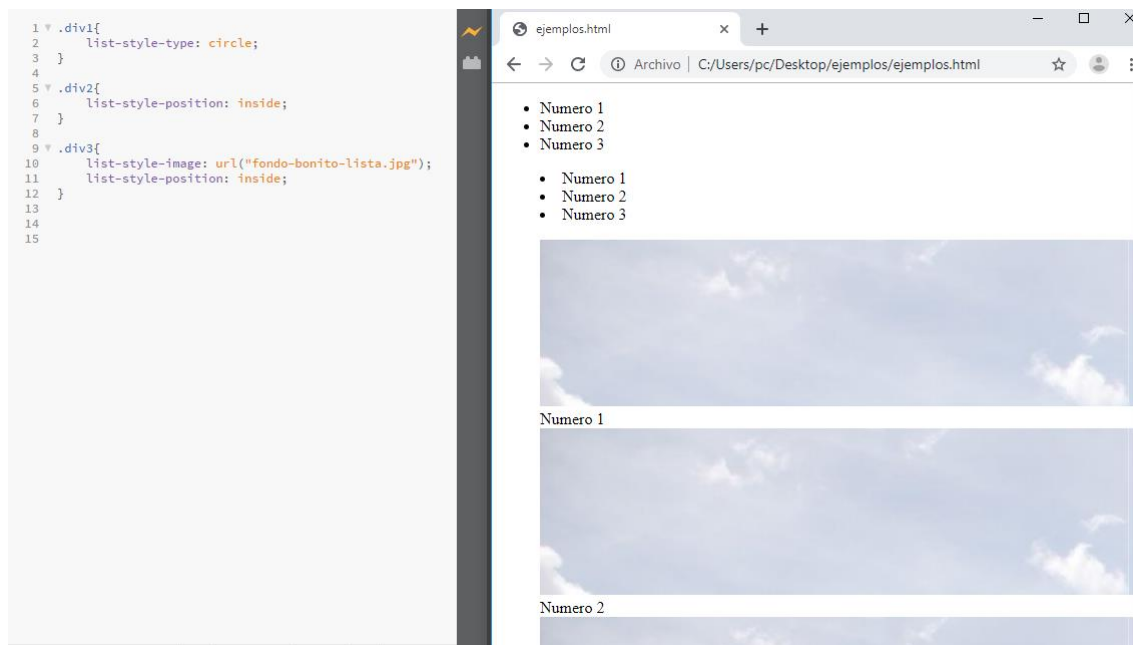
List

La propiedad “list” se utiliza siempre en compañía para modificar las listas de html, se puede utilizar con las propiedades “style-type”, para modificar el estilo de los incisos, “style-image”, para agregar una imagen entre cada inciso de la lista y “style-position”, para colocar una imagen entre cada inciso de la lista; cómo podemos notar, siempre va acompañada de la propiedad “style”. Un ejemplo de cada una de estas propiedades sería:

```
list-style-type: circle;
```

```
list-style-image: url(“imagenbonita.jpg”);
```

```
list-style-position: inside;
```



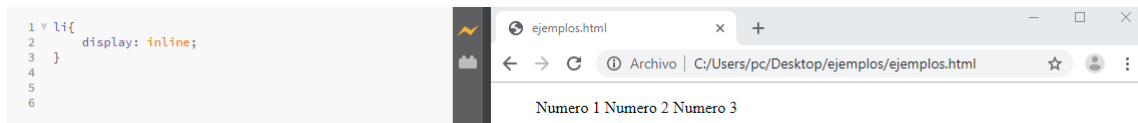
Como podemos notar, utilizamos las tres propiedades anteriormente mencionadas; para definir el estilo de la lista como circular (los incisos), luego para cambiarle la posición a la siguiente línea, y luego para poner una imagen (es el mismo fondo utilizado anteriormente, pero editado para que se acople al espacio de la página) entre cada valor de la lista.

Display

La propiedad “display” es la propiedad más importante en CSS para el control del diseño. Esta especifica cómo un elemento será mostrado y también si este se muestra. Antes de explicar la propiedad, se tiene que decir que cada elemento en html tiene una posición a la hora de mostrarse en pantalla.

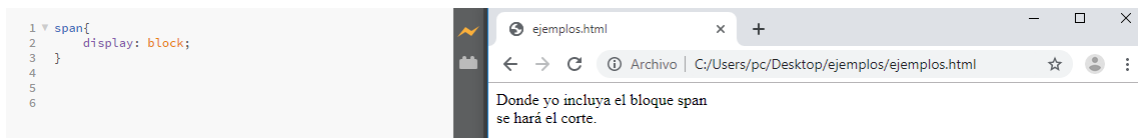
Las siguientes etiquetas siempre comenzarán en una nueva línea y se tomará siempre toda la anchura disponible: <div>, <h1> - <h6>, <p>, <form>, <header>, <footer>, <section>, y tenemos las siguientes que estarán siempre en la posición donde se pongan (o sea, si yo ponga estas etiquetas en medio de una letra, ahí estarán), y nunca empiezan una nueva línea, a menos que se indique, por supuesto, con un salto de línea: , <a>, ; estas etiquetas sólo utilizarán la anchura necesaria para mostrarse por completo.

Sabiendo esto, se queda claro que cada elemento tiene una posición de visualización predeterminada (un valor, para ser más exactos), pero este valor se puede anular, ¿cómo? Con la propiedad “display”. Usando esta regla se define que las listas estarán en línea:



Como notamos, esta propiedad se la especificamos a los elementos de las listas, haciendo que estas, en vez de salir con un salto de página pre definido, saldrán de manera lineal.

También se puede utilizar con el valor “block”, haciendo que se bloquee el nuevo posicionamiento de algo, reaccionando de esta manera:



Como notamos, con el valor “block”, hacemos que cada vez que haya un bloque span, se hará un bloqueo de visualización, o sea, que ahí —en el espacio del bloque- no se podrá ingresar más que lo que está en el bloque.

Ahora, existen dos posibilidades para hacer lo mismo, esconder algo, la propiedad, igualmente “display”, pero con el valor “none” y la propiedad “visibility” con el valor “hidden”, o sea:

display: none;

visibility: hidden;

Ambas logran lo mismo, que un elemento no aparezca, pero, el primero hará que no se ocupe ningún espacio, en cambio, si se encuentra escondido, con la segunda propiedad, si ocupará el espacio normal sin la propiedad, pero el elemento no estará a la vista.

Box-Sizing

Esta propiedad nos permite incluir el relleno y el borde en la altura y anchura total de un elemento. Por default, la anchura y la altura se calculan de esta manera:

Anchura = Anchura + relleno + borde.

Altura = Altura + relleno + borde.

Esto significa que, cuando le das un valor a la altura o anchura de un elemento, se verá más grande, debido a que para calcular ese valor no toma en cuenta solamente la altura, o anchura, sino también el relleno y el borde. Este problema se resuelve con la propiedad “box-sizing”.

Utilizando el valor “border-box”, se cumple esto, de la forma:

```
box-sizing: border-box;
```

Por lo general, esta propiedad se aplica a toda la página debido a que el resultado utilizándola es mucho mejor. Así que, es una recomendación, aplicarlo a todo el documento, o sea,

```
*{  
  
box-sizing: border-box;  
  
}
```

Por supuesto que posee otros valores, pero este es el más utilizado y recomendado.

Position

Esta será la última propiedad que se explicará en este manual; está propiedad ya la hemos utilizado, pero como acompañamiento a otra propiedad, pero, se prefiere tratar al final para tratarla por completo todos sus valores. La propiedad, como sugiere su traducción, es para elegir la posición de algún elemento.

El valor más usado es “fixed”, de la forma:

```
position: fixed;
```

Esto hace que la posición sea relativa a la ventana gráfica, o sea que siempre se mantendrá en el mismo lugar, incluso si la página es movida con el scroll.

Esto aplica para imágenes principalmente, pero se puede utilizar de una manera muy buena con bloques <div>.

También tenemos el valor “absolute”, de la forma:

```
position: absolute;
```

Lo que hace es que el elemento sea posicionado con relación con el último elemento, pero, si no tiene un “antepasado”, o sea, no hay un elemento detrás de esta, usará el <body> del documento, lo que hará que se mueva con el scroll.

Tenemos luego el valor “sticky”, de la forma:

```
position: sticky;
```

Este valor hace que el elemento sea posicionado según el scroll. Primero se posiciona relativamente (otro valor de position, que se puede comprobar con esta posición, el valor relativo, o sea, con la propiedad “relative” sería la posición primera del elemento con el valor “sticky”) y luego, cuando el scroll se baje (o suba) hasta una posición donde en teoría el elemento no se muestra, tomará un comportamiento “fixed” y se mantendrá en la última posición, o sea, en lo más debajo de la página o en lo más por encima de la página si se sube o se baja, respectivamente.

Aquí también se hablará de la propiedad “z-index”, la cual sirve para superponer elementos, dándoles un “orden” o jerarquía controlado por el programador, o sea, si yo utilizo una imagen, que se estaría poniendo por encima de un texto, pero la quiero tener de fondo, puedo utilizar esta propiedad para que la imagen sea superpuesta por el texto, ¿cómo? Así:

```
z-index: -1;
```

Aquí se estará diciendo que la imagen estará por debajo.

Con esto, se termina la explicación de las propiedades, ahora, te propongo dos ejercicios para la utilización y combinación de todas estas propiedades, aparte de un pequeño uso de html para la aplicación de las hojas de estilo, por supuesto.

1. Deseo crear una página web en la que tenga de fondo de pantalla una imagen de un paisaje. La página web tendrá una cabecera con un título (PRIMERA TAREA DEL MANUAL) de un color que se vea bien. La página ocupará un espacio de 1000px. Deseo también tener un div en el medio de la página donde salgan los datos de cada persona que haga este ejercicio, o sea, nombre, apellido y correo electrónico.

2. Deseo un sitio web que tenga dos páginas, un inicio y una información de contacto. El inicio lo deseo simple. Un banner en el tope con imágenes alusivas a una tienda de computadoras, un fondo de pantalla alusivo al mismo tema, y un pie de página con otro color, donde salga un nombre de tienda (sean creativos). Los demás detalles los dejo a gusto personal, pero, no dejen la página en blanco 😊. La página de contactos tendrá un recuadro pequeño, en la posición elegida al gusto, con una información de contacto ficticia, repitiendo el mismo banner y el mismo pie de página, además de la misma imagen de fondo.

JavaScript

JavaScript es un lenguaje de tipo script, interpretado, y cuya plataforma de ejecución más frecuente son los navegadores. JavaScript también puede correr fuera de los navegadores gracias a entornos como Node JS, que permite ejecutar código JavaScript para tareas que requieran integración de módulos o ejecución de código en servidores.



¿Para qué nos sirve JavaScript en el desarrollo Web?

En la actualidad JavaScript es lenguaje más utilizado en el mundo del desarrollo, ya que está presente en prácticamente todo el internet. Hasta el momento sabes que con los otros dos lenguajes pilares de la Web, HTML y CSS, podíamos estructurar y estilizar respectivamente, páginas web, ahora con JavaScript podemos darle interactividad.

A continuación, se mencionan los usos más frecuentes que se le da a JavaScript:

- **Modificación de la estructura.** Con JS podemos modificar de forma dinámica la estructura y propiedades de un sitio Web. Podemos agregar y eliminar atributos de etiquetas HTML e incluso las etiquetas mismas.
- **Procesos asíncronos.** Una de las características de JavaScript es poder realizar procesos independientes al flujo principal. JavaScript trabaja con un único hilo de ejecución y el asincronismo viene a cumplir con la función de ejecutarse procesos paralelos sin detener el flujo principal y único. La mayoría de ejecuciones de esta naturaleza se realizan cuando se solicitan datos a servidores y se cargan a la página modificando su estructura, esto se da, por ejemplo, cuando una aplicación consigue información sin recargar el sitio.

- **Eventos.** Con JavaScript podemos asignar a eventos a elementos HTML, y que cuando ocurra cierto escenario, se ejecuten indicaciones para realizar algún cambio u otra actividad.
- **Validaciones.** JavaScript se utiliza con mucha frecuencia para validar métodos de entrada de información desde el área del cliente. Estas pueden ser consultas asíncronas o verificaciones a partir del contenido HTML.

¿Qué se va a abarcar?

En esta ocasión omitiremos los detalles básicos del lenguaje (que podrían ser de mayor utilidad para otras tareas) para enfocarnos en cómo usar de forma práctica el lenguaje para tareas comunes en el desarrollo Web.

Entre lo que se va a explicar se encuentran las distintas formas de incorporar código JavaScript a un documento HTML, modificar elementos seleccionándolos desde el DOM, asignar un evento a un elemento HTML.

¿Cómo agregar JavaScript?

Básicamente, podemos agregar de dos formas: directamente en el HTML con la ayuda de etiquetas `<script></script>` o bien con un archivo externo:


```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta http-equiv="X-UA-Compatible" content="ie=edge">
  <title>Document</title>
</head>

<body>
  <div class="box" id="box"></div>

  <script>
    var box = document.getElementById("box");
    box.style.height = "400px";
    box.style.width = "400px";
    box.style.background = "green";
  </script>
</body>
</html>
```

```


JS prueba.js ▸ ...
1 // prueba.js
2
3
4 var box = document.getElementById("box");
5 box.style.height = "400px";
6 box.style.width = "400px";
7 box.style.background = "green";

<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta http-equiv="X-UA-Compatible" content="ie=edge">
  <title>Document</title>
</head>

<body>
  <div class="box" id="box"></div>

  <script src="prueba.js"></script>
</body>
</html>

```



Tipos de datos en JavaScript

Una característica de JavaScript es ser un lenguaje débilmente tipado, lo que quiere decir que no maneja un tipo de valor estático en las variables. Una variable puede recibir cualquiera de los tipos de datos que veremos a continuación, y estas variables pueden cambiar el tipo de valor que almacenan a lo largo de proceso de ejecución.

Tipos numéricos:

Tenemos valores de enteros y decimales, aunque JS sólo maneja un tipo de numérico:

```

var a = 5

var b = 4

Console.log(a + b) // Debe mostrar 9 en la consola

```

```
var a = 5.44

var b = 4

console.log(a + b) // Debe mostrar 9.440000000000001 en la consola
```

Los valores de tipo **cadena** funcionan tanto con comillas simples (') como con comillas dobles; aquí no existe el tipo carácter:

```
var name = "Julio "
var lastName = 'García'

console.log(name + lastName) // Ambas se tratan igual y se muestra "Julio García"
```

Los variables de tipo **boolean** aceptan como valores true o false:

```
var val = (20 == 20) // Se asigna verdadero

var val2 = (10 > 30) // Se asigna falso
```

De forma superficial (será explicado más adelante) un **array** se estructura de la siguiente forma:

```
var friends = ["Mark", "Yoko", "Esteban"]
```

Los **objetos** en JavaScript se escriben con llaves ({}) y en pares a modo clave-valor:

```
var auto = { marca : "Toyota", modelo : "Hilux", year : 2014 }

console.log("Hola, tengo una " + marca + " " + modelo + " del " + year)
```

Si declaramos una variable sin asignarle un valor, esta automáticamente recibe el valor **undefined**. Existe también el valor **null**, que a nivel práctico es similar a **undefined**, con la diferencia que si el primero, curiosamente, es asignado a una variable, se le reconoce como un objeto, aunque su valor sea nada.

```
var b // Su valor es undefined  
  
var k = null // Su valor es null
```

Arrays

Aquí tenemos unos ejemplos que declaran un array:

```
var arr = ["Leonovo", "Dell", "HP", "Acer"]  
  
var arr2 = array("Leonovo", "Dell", "HP", "Acer")
```

Y para acceder a un elemento especificamos la posición:

```
var arr = ["Leonovo", "Dell", "HP", "Acer"]  
  
var arr2 = array("Leonovo", "Dell", "HP", "Acer")  
  
console.log(arr[1]) // Mostrará "Dell"
```

En JavaScript, los arrays son objetos. Una propiedad muy utilizada de ellos es `length`, que nos retorna la cantidad de elementos que contiene la colección y suele sernos útiles en los ciclos.

```
var arr = ["Leonovo", "Dell", "HP", "Acer"]  
  
var arr2 = array("Leonovo", "Dell", "HP", "Acer")  
  
console.log(arr.length) // Mostrará 4
```

Algunos métodos útiles de los arrays:

- `pop()` remueve el último elemento del array
- `push()` agrega un nuevo elemento al array
- `slice()` retornará un array recortado desde la posición que le especifiquemos.

Estructuras condicionales

Si ya estás familiarizado con lenguajes de programación influenciados por C, no necesitarás prestar mucha atención a esta parte, pues la estructura en JavaScript no varía.

```
var edad = 17

if(edad < 18)
{
    console.log("Eres menor de edad")
}else{
    console.log("Eres mayor de edad")
}
```

Aquí hacemos el uso de tradicional del bloque **if**, que ejecutará la siguiente sentencia en caso de que se apruebe la condición; el bloque **else** ejecutará las sentencias en caso de que la condición resulte falsa.

```
var edad = 17

if(edad < 18)
{
    console.log("Eres menor de edad")
}else if(edad == 17){
    console.log("Casi eres mayor de edad")
} |
else{
    console.log("Eres mayor de edad")
}
```

También podemos agregar otro comprobador agregando **if else**, seguido de la evaluación. Podemos agregar más de un segmento de **if else**, y en caso de cumplirse ninguna de las condiciones, se ejecutan las sentencias del bloque **else**.

```

switch(value)
{
  case 1:
    console("Hola")
    break
  case 2:
    console.log("Hola a todos")
    break
  case 3:
    console.log("Qué tal")
    break
  default:
}

```

El bloque switch nos permite pasar un valor, y partir de él encontrar coincidencia con los valores candidatos. En caso de no cumplirse alguno, se procede a lo preparado en default. La sentencia break es necesaria, esta pone fin a las evaluaciones condicionales y nos evitamos resultados inesperados.

```

var value = "Frank"

switch(value)
{
  case "Frank":
    console("Hola")
    break
  case "Carla":
    console.log("Hola a todos")
    break
  case "Fred":
    console.log("Qué tal")
    break
  default:
    console.log("Adiós")
}

```

En este ejemplo podemos ver que es posible utilizar cadenas de caracteres como valores candidatos en la estructura switch.

Pero, ¿qué tal si queremos una sentencias que se apliquen para más de una coincidencia? Podemos hacer lo siguiente:

```
switch(value)
{
  case "Frank":
  case "Carla":
    console.log("Hola a todos")
    break
  case "Fred":
    console.log("Qué tal")
    break
  default:
    console.log("Adiós")
}
```

Los valores que coincidan tanto con “Frank” como con “Carla” tendrán como resultado la ejecución de la misma sentencia.

Estructuras repetitivas

Como ocurre con el tema anterior, las estructuras repetitivas o los ciclos no varían respecto a los demás lenguajes influenciados por C, por lo que la tarea de aprenderlos se hace más fácil.

Ciclo for

```
for(i = 0; i < 10 ; i++)
{
  console.log(i)
}
```

Esta es la forma tradicional de estructurar un ciclo **for**. Sin embargo, existen más variantes, como las que veremos a continuación.

El ciclo **for/in** lo utilizamos

para recorrer los atributos de una clase:

```
var mike = {name: "Mike", lastName: "Smith", age : 24}

for(pro in mike)
{
  console.log(pro)
}
```

El ciclo **for/of** nos permite iterar en estructuras como Arrays, Strings, Maps, NodeLists, por ejemplo. Aquí una demostración con una cadena de caracteres:

```
var texto = "Gilberto"

for(txt of texto)
{
  console.log(txt)
}
```

Ciclo while

El ciclo while trabaja con una condición que siempre que sea verdadera ejecutará la iteración.

```
while( a < 10)
{
  console.log(a)
  a++;
}
```

El ciclo do while, sigue la misma mecánica, con la diferencia que en este hay una iteración que siempre se realizará (la inicial).

```
do{
  console.log(a)
  a++;
}while(a < 9)
```


Funciones

Las funciones son segmentos de código reutilizable que siempre ha permitido a los desarrolladores generar un código más consistente, mejor comprensible y modularizado, Ahora veremos la forma de trabajar con funciones en JavaScript.

Las funciones se pueden definir como variables, he aquí un ejemplo con una constante:

```
const multiplicar = function(a, b){ // Definida como constante
  return a * b
}

function multiplicar(a, b) // Forma tradicional
{
  return a * b
}
```

Arrow Functions

Existe otra notación para escribir funciones, esta consiste en omitir la palabra reservada function y poner un signo => (no confundir con >=).

```
const sum = (a, b) =>{
  return a * b
}
```

Los símbolos => viene después de la lista de parámetros, y lo que le sigue es el cuerpo de la función.

```
doble = (x) => {
  return x * 2;
}

doble2 = x =>{
  return x * 2
}
```

```
}
```

Si se utiliza un parámetro, se pueden omitir los paréntesis, como podemos observar en arriba en el código.

```
saludo = () => {  
  console.log("Hola a todos")  
}
```

En caso de no esperar parámetros sólo se escriben los paréntesis.

Clases

Las clases en JavaScript se declaran con la palabra reservada **class** seguido del nombre y cuerpo de la clase. He preparado un ejemplo con una clase llama persona, que fue de la siguiente manera:

```
class Persona{  
  constructor(nombre, apellido, edad, altura)  
  {  
    this.nombre = nombre  
    this.apellido = apellido  
    this.edad = edad  
    this.altura = altura  
  }  
  
  saludar(){  
    console.log(`Hola, soy ${this.nombre} ${this.apellido}`)  
  }  
  
  decirEdad(){  
    console.log(`Actualmente tengo ${this.edad} años`)  
  }  
  
  decirAltura(){  
    console.log(`Yo soy ${this.altura} M`)  
  }  
}
```

El constructor es la función que inicializa los atributos de la clase. Los atributos se pueden declarar directamente en el constructor.

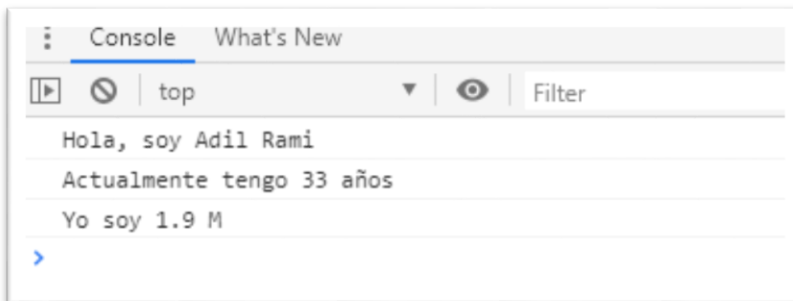
Ahora creemos una instancia:

```
var adil = new Persona('Adil', 'Rami', 33, 1.90)
```

Ejecutemos sus funciones:

```
adil.saludar()  
adil.decirEdad()  
adil.decirAltura()
```

El resultado en consola:



Seleccionando un elemento con JavaScript

Para poder seleccionar un elemento es necesario asignarle una propiedad identificativa, por lo general se escoge un id:

```
<div class="box" id="box"></div>
```

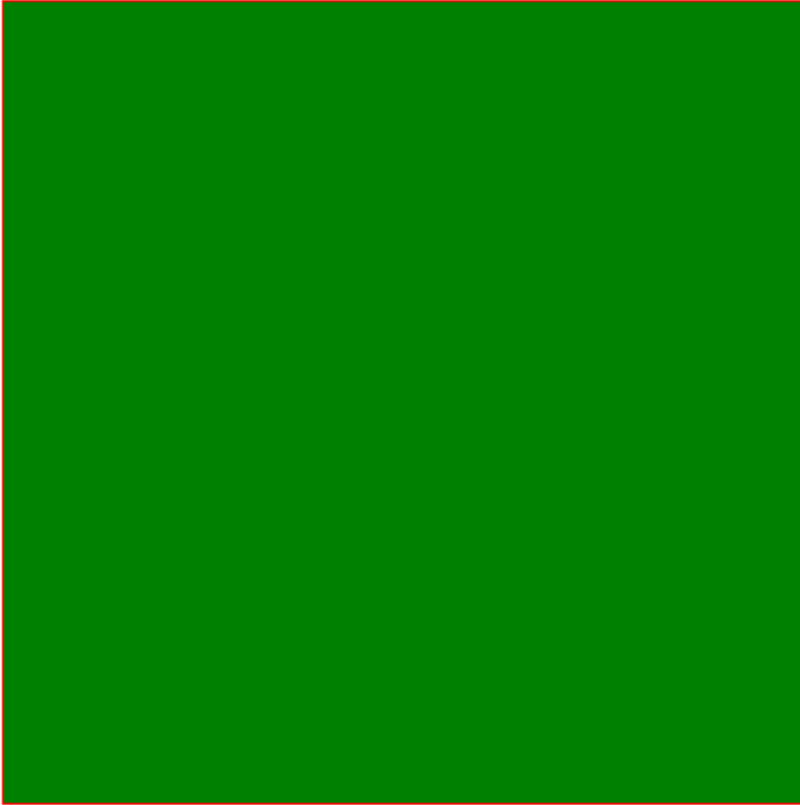
Ahora podemos seleccionarlo desde un archivo JavaScript:

```
var box = document.getElementById("box");  
box.style.height = "400px";  
box.style.width = "400px";  
box.style.background = "green";  
box.style.border = "solid red 1px";
```

En la asignación de la variable “box” escribimos “document” para hacer referencia al archivo desde el cual es utilizado el archivo. Luego, con `getElementById` especificamos el id del elemento que queremos obtener.

Ahora, la variable `box` representa al elemento que teníamos en el archivo HTML. Más que variable, “box” es un objeto, y como objeto contiene propiedades. La propiedad “style”, como podemos ver en la captura nos permite asignarle estilos como si bien se tratara de un archivo CSS, sólo procura usar un editor con autocompletado para hacer esto más fácil. Te aparecerán las propiedades de estilos, y las reconocerás si ya dominas CSS.

Ahora veámoslo en el navegador:




Los estilos se han aplicado a como lo especificamos. También podemos usar la función `getElementsByClassName`, sólo que este nos asociará todos los elementos que contengan la clase especificada.

Ya probamos con estilos, ¿te parece si probamos ahora con eventos?

Eventos con JavaScript

Para este paso podemos utilizar los eventos de HTML, en los cuales se especifica la función de JavaScript que se ejecutará cuando el evento se detecte.

```
10 <body>
11   <div class="box" id="box" on</div>
12     <script src="prueba.js"></script>
13   </body>
14 </html>
```



Como podemos ver, tenemos gran cantidad de eventos escribiendo tan solo las iniciales “on”, yo escogeré onMouseOver que sería un equivalente a :hover en CSS.

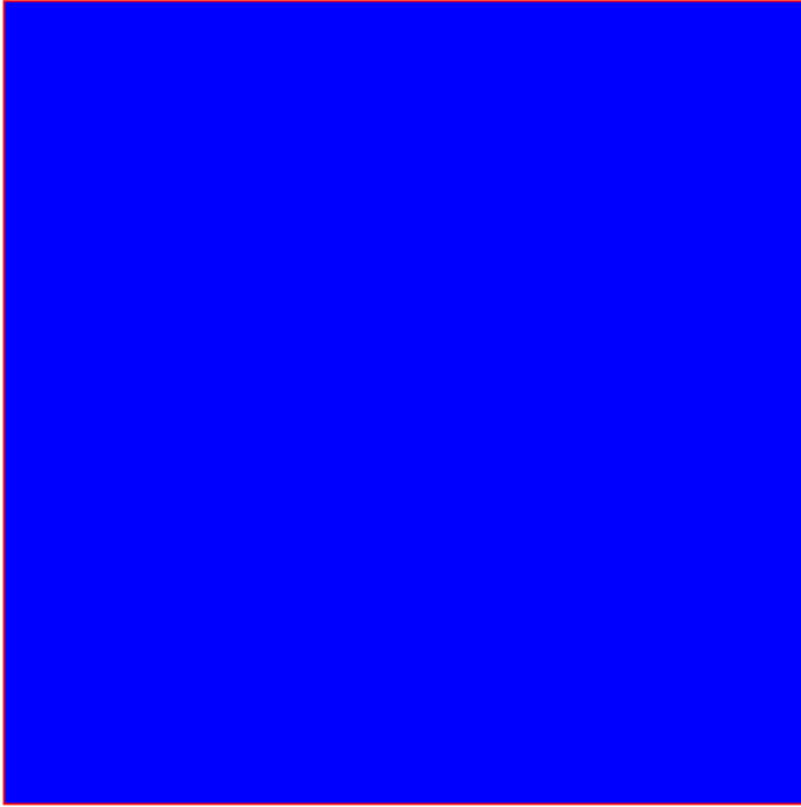
```
<body>
  <div class="box" id="box" onmouseover="volverAzul()"></div>
  <script src="prueba.js"></script>
</body>
</html>
```

Una vez especificada la función, procedemos a codificarla en el archivo JS:

```
box.style.background = "green";
box.style.border = "solid red 1px";

function volverAzul(){
  box.style.background = "blue";
}
```

Veamos qué pasa al poner el mouse sobre el cuadro:



¡Puedes comprobarlo tú mismo si quieres!

Ahora, probemos otra alternativa. Usando `addEventListener`:

```
/* function volverAzul(){  
|   box.style.background = "blue";  
| } */  
  
box.addEventListener("mouseover", () =>{  
|   box.style.background = "blue";  
| });
```

El primer parámetro es el tipo de evento que esperamos, que, si cuentas con un editor con autocompletado, no se te hará difícil este paso.

El resultado es completamente el mismo que el anterior ¡Pruébalo!

JQuery

JQuery es una librería de JavaScript que es ampliamente utilizada en el desarrollo Web. JQuery simplifica el uso de JavaScript para determinadas tareas. JQuery es principalmente utilizado para el manejo la manipulación del DOM, las animaciones y las tareas asíncronas (con AJAX).

¿Cómo incorporar JQuery?

Básicamente hay dos formas de agregar, la primera descargar el script de la librería, y la segunda es incorporarla de forma remota utilizando algún enlace, como el que te dejo a continuación:

```
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>
```

Lo más práctico es situarlo justo antes de cerrar la etiqueta body de nuestro archivo HTML.

Sintaxis

Básicamente, la sintaxis de uso de JQuery se resumiría a esto: **\$(selector).accion()**

Ejemplos a continuación:

```
<script>
  $("p").hide() // Ocultará todos los párrafos
  $(".text").hide() // Oculta todos los elementos con la clase test
  $("#test").hide() // Oculta todos los elementos con el id test
</script>
```


Selectores

```
<script>
  $(document).ready(function(){
    $("button").click(=>{
      $("p").hide();
    })
  })
</script>
```

En este ejemplo, una vez que el documento haya terminado de cargar haremos que al presionar cualquier botón todos los párrafos se ocultarán.

```
$(document).ready(function(){
  $("button").click(=>{
    $(".container").hide();
  })
})
```

Aquí se ocultarán todos los elementos con la clase “container”.

Más ejemplos de selección:

```
$("#") // Selecciona todos los elementos
$("#this") // Referencia al elemento actual
$("#p.intro") // Selecciona todos los párrafos con la clase intro
$("#p:first") // Selecciona el primer párrafo
$("#[href]") // Selecciona todos los elementos con el atributo href
$("#[type='text']") // Selecciona todos los elementos con un atributo type igual a text
$("#a[tarjet='_blank']") // Selecciona todos los elementos <a> que tengan un atributo
                        // tarjet igual _blank
```

Eventos con JQuery

El evento más básico, es el del click, este se define de la siguiente manera:

```
$("#p").click();
```

El siguiente paso es definir qué pasará al clickear, por ende, se tiene que definir una función, por ejemplo:

```
$("#p").click(function(){
```

-Las acciones se ponen acá.

```
});
```

Un ejemplo más concreto sería el de ocultar el elemento que se clickea, el cual tendría la forma:

```
$("#p").click(function(){
```

```
    $(this).hide();
```

```
});
```

Ahí se especifica que el elemento "p" clickeado, se ocultará.

Tenemos también los eventos "mouseenter()" y "mouseleave()", que como la traducción sugiere, estos eventos se activarán cuando el puntero del mouse entre en el elemento html y cuando este salga, respectivamente, con la forma:

```
$("#p1").mouseenter(function(){
```

```
    alert("Entraste al elemento p1");
```

```
});
```

Y, para el otro evento, sólo se reemplaza la palabra "mouseenter()" por "mouseleave()".

También tenemos el evento "hover()", el cual toma dos funciones, siendo la combinación de las dos anteriores mencionadas, con la forma:

```
$("#p1").hover(function(){  
    alert("Entraste al elemento p1");  
},  
function(){  
    alert("Adiós, te vas del element p1");  
});
```

Aquí estamos combinando los dos eventos con simplemente el elemento "hover()"; como se puede observar, lo primero corresponde al evento "mouseenter()" y lo siguiente al evento "mouseleave()".

Efectos con JQuery

Hide/Show

Tenemos los dos eventos más conocidos, "Hide" y "Show", que como la traducción lo dice, sirven para ocultar o mostrar un elemento, con la forma:

```
$("#hide").click(function(){  
    $("#p").hide();  
});
```

Donde se especifica que el elemento "p" se ocultará al suceder el evento "click()" sobre el elemento del ID "hide". Show se utiliza de la misma manera.

También se puede crear una palanca, o sea, un efecto "toggle", para alternar entre "show" y "hide", con la forma:

```
$("#button").click(function(){  
    $("#p").toggle();  
});
```

Donde se alternará entre ocultar y mostrar los párrafos "p" cuando se clickee el botón. También existen valores para editar un poco esta palanca predeterminada, donde:

```
$("#p").toggle("velocidad", "callback");
```

“Velocidad” es reemplazable por “slow”, “fast” y milisegundos, para definir la velocidad del estilo, y “callback” es una función que se ejecuta cuando el efecto “toggle” se completa.

Fading

Con este efecto, se puede desvanecer un elemento dentro y fuera de la visibilidad de el usuario.

Se utiliza de la forma:

```
$("#button").click(function(){  
  
    $("#p").fadeIn();  
  
});
```

Donde, “fadeIn()” es reemplazable por “fadeOut()”, “fadeToggle()” y “fadeTo()”, donde, dentro del paréntesis se pueden sugerir velocidades (“slow”) o (“fast”), o escribir en milisegundos (5000), a excepción de el “fadeTo()”, donde, dentro del paréntesis, se especifica, la velocidad, y, separado por una coma, el valor de desvanecimiento, por ejemplo: fadeTo(“slow”, 0.7);.

Te animo a probar todas las variantes del efecto para dominarlos.

Sliding

Este efecto se utiliza para deslizar algo hacia abajo o hacia arriba u ambos lados, o sea, “toggle()”.

La sintaxis del efecto es:

```
$("#panel").slideDown();
```

Donde, panel es reemplazable por cualquier otra etiqueta u id y “slideDown()” por “slideUp()” y “slideToggle()”. Dentro del paréntesis, igualmente, se puede definir la velocidad con los valores “slow”, “fast” o milisegundos, una coma (,) y la función “callback”.

Callback

Se ha mencionado varias veces en los efectos anteriores “la función callback”, pero, ¿qué es esto? Este espacio es para otra función que se activará exactamente al finalizar el efecto anterior, por ejemplo:

```
$("#button").click(function(){  
  
    $("#p").hide("slow", function(){  
  
        alert("El párrafo ahora está oculto");  
  
    });  
  
});
```

Como se puede apreciar, luego del efecto “hide()”, ponemos la velocidad “slow”, y, reemplazando al espacio “callback”, denotamos que viene otra función, en este caso, una alerta que dirá que el párrafo está oculto. Luego se cierran ambas funciones, la original, o sea, “click()” y luego el “callback”.

Referencias

W3schools.com. (s.f.). CSS width Property, recuperado de:
https://www.w3schools.com/cssref/pr_dim_width.asp

W3schools.com. (s.f.). CSS height Property, recuperado de:
https://www.w3schools.com/cssref/pr_dim_height.asp

W3schools.com. (s.f.). CSS Padding, recuperado de:
https://www.w3schools.com/css/css_padding.asp

W3schools.com. (s.f.). CSS Borders, recuperado de:
https://www.w3schools.com/css/css_border.asp

W3schools.com. (s.f.). CSS Margins , recuperado de:
https://www.w3schools.com/css/css_margin.asp

W3schools.com. (s.f.). Propiedad Background, recuperado de:
<https://uniwebsidad.com/libros/referencia-css2/background>

W3schools.com. (s.f.). Tutorial CSS3: El uso de reglas, recuperado de:
<https://www.dariobf.com/tutorial-css3-regla-css/>

W3schools.com. (s.f.). CSS Selector Reference, recuperado de:
https://www.w3schools.com/cssref/css_selectors.asp

Definicion.de. (s.f.). Definición de HTML, recuperado de: <https://definicion.de/html/>

Diseño y elaboración de páginas web.blogspot.com (s.f.). Partes de un documento HTML, recuperado de: <http://disenoyelaboraciondepaginasweb.blogspot.com/2012/04/partes-de-un-documento-html.html>

Developer.mozilla.org (s.f.). Link, recuperado de:
<https://developer.mozilla.org/es/docs/Web/HTML/Elemento/link>

Mclibre.org (s.f.). Favicon, recuperado de:
<http://www.mclibre.org/consultar/htmlcss/html/html-favicon.html>

Lenguajehtml.com (s.f.). Etiquetas HTML de texto, recuperado de:
<https://lenguajehtml.com/p/html/semantica/etiquetas-html-de-texto>

Developer.mozilla.org (s.f). Images in HTML, recuperado de:
https://developer.mozilla.org/es/docs/Learn/HTML/Multimedia_and_embedding/Images_in_HTML

Htmlquick.com (s.f). Tables, recuperado de:
<http://www.htmlquick.com/es/tutorials/tables.html>

Developer.mozilla.org (s.f). Elemento Div, recuperado de:
<https://developer.mozilla.org/es/docs/Web/HTML/Elemento/div>

Wikipedia.org (s.f). Form, recuperado de:
[https://es.wikipedia.org/wiki/Form_\(etiqueta_HTML\)](https://es.wikipedia.org/wiki/Form_(etiqueta_HTML))

Developer.mozilla.org (s.f). Button, recuperado de:
<https://developer.mozilla.org/es/docs/Web/HTML/Elemento/button>

Developer.mozilla.org (s.f). Input, recuperado de:
<https://developer.mozilla.org/es/docs/Web/HTML/Elemento/input>

Manualweb.net (s.f). Listas, recuperado de: <http://www.manualweb.net/html/listas-html/>

Manualweb.net (s.f). Enlaces, recuperado de: <http://www.manualweb.net/html/enlaces-html/>

W3schools.com. (s.f.). JavaScript Tutorial. Recuperado de:
<https://www.w3schools.com/js/default.asp>

W3schools.com. (s.f.). JQuery Tutorial. Recuperado de:
<https://www.w3schools.com/jquery/default.asp>

Haverbeke, M. (s. f.). Eloquent JavaScript 3ra Edición. Recuperado de:
<https://eloquentjavascript.net/>