



HTML y CSS

Argentina Programa 4.0
Programación Web Full Stack
Tramo II

MATERIAL ADICIONAL DE LECTURA

Profesores: **Sabrina Castagno** y **Maximiliano Firtman**

v.1.0: Junio 2017

v.2.3: Julio 2023

Autores: Sabrina Castagno, ITMaster Academy

Edición Revisada: Maximiliano Firtman, ITMaster Academy



Este contenido se distribuye bajo licencia Creative Commons 3.0 con atribución (<https://creativecommons.org/licenses/by/3.0/>)

Se acepta su copia y redistribución en cualquier medio y formato y adaptarlo en cualquier forma (remixado, transformado y construir sobre él) para cualquier propósito, incluso comercial bajo una única condición: la atribución y crédito a sus autores y a la aclaración de cambios realizados en caso de haberse realizado

INDICE

INDICE.....	3
<i>¿QUÉ ES EL DISEÑO WEB?</i>	6
EMPEZAR CON HTML Y CSS	7
<i>¿QUÉ ES HTML?</i>	7
<i>¿Qué es una etiqueta o marca?</i>	7
<i>¿Cómo hago mi primer página?</i>	8
<i>Poniéndole nombre a los archivos HTML</i>	9
<i>¿Qué es el Dominio?</i>	9
CREANDO NUESTRO PRIMER ARCHIVO	10
<i>¿Dónde lo guardo?</i>	11
<i>Estructura básica de un archivo HTML</i>	12
PRIMEROS ELEMENTOS DE TEXTO.....	15
<i>Encabezados</i>	15
<i>Párrafos de Texto</i>	15
<i>Atributos</i>	16
PREVISUALIZANDO NUESTRO TRABAJO	16
OTROS ELEMENTOS	18
<i>Comentarios del diseñador</i>	18
<i>Saltos de Línea</i>	18
<i>Listas</i>	19
<i>Elementos de formato de texto</i>	21
<i>Imágenes</i>	22
VÍNCULOS O LINKS.....	23
<i>Atributos de los vínculos</i>	24
<i>Vínculo de Correo Electrónico</i>	25
<i>Anclas Internas</i>	26
EL LENGUAJE CSS	28
<i>Archivo de Estilos Externo</i>	28
<i>Lista de estilos Interna</i>	29
<i>Estilos en línea</i>	30
ENTENDIENDO LAS REGLAS DE ESTILOS.....	31
<i>El Selector</i>	31
PROPIEDADES Y ESTILOS EN CSS.....	33
<i>Familia de Tipografía: font-family</i>	33
<i>Negritas: font-weight</i>	36
<i>Estilos de letra: font-style</i>	36

<i>Tamaño de Letra: font-size</i>	37
<i>Color de frente: color</i>	37
<i>Transformación de Texto: text-transform</i>	39
<i>Alineación de Texto: text-align</i>	39
CONTENEDORES DE CONTENIDO	39
<i>Encabezados Visibles</i>	40
<i>Zona de Navegación</i>	41
<i>Pie de Página</i>	41
<i>Sección</i>	42
<i>Artículos o Items</i>	42
<i>Contenedor Principal</i>	43
<i>Divisor Genérico</i>	44
TRABAJANDO CON BLOQUES EN CSS	45
<i>Propiedades de Caja</i>	45
<i>Propiedades del fondo de la caja</i>	51
ORGANIZACIÓN DE ELEMENTOS CON FLEXBOX	55
<i>Cambiando la dirección de los elementos</i>	56
<i>Espaciado entre elementos</i>	58
<i>Generador online de Flexbox</i>	61
DÁNDOLE ESTILOS A LOS VÍNCULOS	61
<i>Usando pseudo-clases</i>	62
<i>Quitar el subrayado</i>	63
FORMULARIOS.....	64
ENTENDIENDO EL MODELO CLIENTE-SERVIDOR	64
¿QUÉ ES UN FORMULARIO?	64
COMPONENTES BÁSICOS	66
<i>Campos de Ingreso de Texto</i>	66
<i>Etiquetas</i>	66
<i>El Contenedor de Formulario</i>	69
<i>Botones de Envío</i>	70
COMPONENTES AVANZADOS.....	72
<i>Campo de Correo Electrónico</i>	72
<i>Campo de Dirección Web</i>	73
<i>Campo Numérico</i>	74
<i>Campo de Teléfono</i>	74
VALIDACIÓN Y OPCIONES AVANZADAS.....	75
<i>Ayuda de Campo de Formulario</i>	75
<i>Campos Obligatorios</i>	76
<i>Datos Múltiples</i>	76
CAMPOS DE SELECCIÓN NATIVOS	77
<i>Campos de Fecha y Hora</i>	77

<i>Campos de Rango o Deslizadores</i>	78
<i>Campos de Color</i>	79
<i>Campos de Búsqueda</i>	79
<i>Campos de Texto Multilínea</i>	80
<i>Campos para Adjuntar Archivos o Fotos</i>	80
<i>Casillas de Verificación</i>	81
<i>Botones de Radio</i>	82
<i>Selectores de Lista</i>	82
ELEMENTOS NO INTERACTIVOS	84
<i>Medidor</i>	84
<i>Barra de Progreso</i>	85
AGRUPANDO SECCIONES	86
DISEÑANDO FORMULARIOS	86
<i>Pseudo-clases de validación</i>	87
IMÁGENES Y VIDEOS	89
INCOPORANDO IMÁGENES	89
<i>Repaso de Sintaxis img</i>	89
<i>Formatos de Gráficos</i>	90
<i>El formato SVG</i>	92
TRABAJANDO CON FIGURAS Y EPÍGRAFES	96
MAPAS DE IMÁGENES	98
EMBEBER OTRO CONTENIDO HTML CON IFRAMES	100
INCLUYENDO CONTENIDO MULTIMEDIA	102
<i>Reproduciendo Audio</i>	102
<i>Reproduciendo Video</i>	103
DIBUJANDO CON CANVAS	105
CONCLUYENDO	108

¿QUÉ ES EL DISEÑO WEB?

El diseño web es una profesión que nos permite crear una página web que puede publicarse en Internet y ser accedida por usuarios desde sus computadoras o desde un teléfono celular. Estas páginas web permiten publicar información de una empresa, blogs, ofrecer servicios de venta de productos o servicios, integrarse con redes sociales y hasta crear apps para las tiendas de celulares.

La tarea de un diseñador web comienza aprendiendo un lenguaje llamado HTML. En este curso vamos a usar la última versión que es HTML5. Con este lenguaje aprenderás a comunicarte con el navegador (como Google Chrome) y hacer que el mismo dibuje un contenido como quieras que aparezca.

El lenguaje HTML no está solo y conviene con su pareja: CSS. Cada uno en la vida de una página web cumple un rol fundamental y ambos funcionan en orquesta para hacer que un sitio web funcione. HTML se encarga del contenido (como el texto y las fotos) y de la semántica o significado de cada parte del contenido (por ejemplo, cuál es el título, qué texto es más importante) y CSS se encarga de definir los estilos gráficos (colores, tamaños, márgenes) y la ubicación de los elementos en la pantalla del navegador.

El diseño web es el paso previo a poder empezar a hacer programación web y de web apps, así que si querés ser programador o programadora web, ¡éste también es tu lugar!

Durante este curso podrás crear tu propia página web así que estamos listos para empezar. ¡Qué lo disfrutes!

EMPEZAR CON HTML Y CSS

¿Qué es HTML?

HTML significa, por sus siglas en inglés Lenguaje de Marcas para Hipertexto. El hipertexto es la posibilidad que nos da Internet de poder vincular un documento (texto) con otro, a través de lo que conocemos como vínculos o links, que nos permite navegar entre distintas páginas usando el mouse o el dedo cuando usamos un equipo táctil.

Para empezar es importante entender los siguientes puntos.

1. Con HTML podés construir tu Sitio Web
2. Para poder trabajar con HTML sólo necesitás un navegador (Microsoft Edge o Google Chrome por ejemplo)
3. HTML funciona con marcas, las cuales son etiquetas que forman parte del lenguaje con el cual tenemos que comunicarnos para que el navegador haga lo que nosotros queremos.

¿Qué es una etiqueta o marca?

Una etiqueta en general está compuesta de la siguiente forma

```
<etiqueta> </etiqueta>
```

Vemos que hay dos etiquetas, una se llama de apertura (la que usa `<>` y el nombre) y otra de cierre (la que usa `</>`) y en el medio se coloca el contenido de dicha marca o etiqueta.

Algunas etiquetas no tienen cierre, entonces se escriben de la siguiente manera:

```
<etiqueta>
```

¿Cómo hago mi primer página?

Lo vas a hacer con un editor de texto. Es un programa que te permite escribir básicamente en cualquier lenguaje o código que quieras.

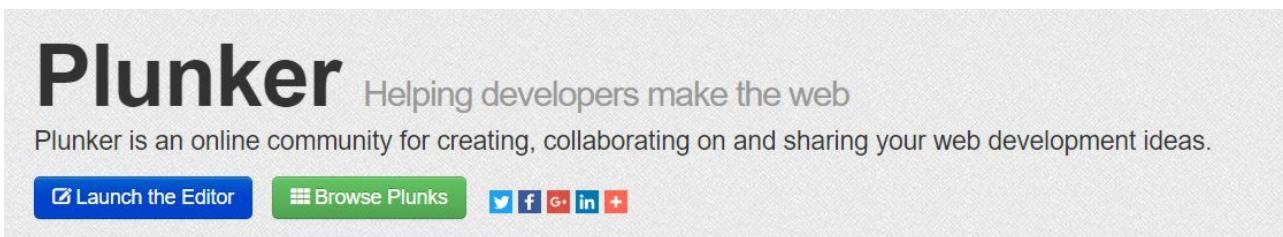
Un archivo que utilice el lenguaje HTML es aquel que tiene un nombre y la extensión (la parte luego del punto del nombre) es html, por ejemplo: mipagina.html.

Es importante que te cuente con qué editores de texto gratuitos podes trabajar que te van a ayudar para crear tu página web, los cuales podrás instalar en una PC o Mac descargándolo de Internet

- Notepad++: <https://notepad-plus-plus.org>
- Sublime Text: <https://www.sublimetext.com>
- Brackets: <http://brackets.io>
- Visual Studio Code: <https://code.visualstudio.com>

Además de poder instalar un editor en tu computadora, también existen editores online, es decir sin bajarlos que podés usar desde un navegador.

Por ejemplo si vas a la página: <https://plnkr.co> te vas a encontrar con el editor online llamado Plunker. Podés trabajar con este editor de manera sencilla. Si querés hacer un archivo nuevo vas a hacer clic en la opción que debajo dice *Launch the Editor* (Lanzar el Editor, en inglés).



The screenshot shows the Plunker homepage. At the top, there's a large logo with the word "Plunker" in a bold, black, sans-serif font, followed by the tagline "Helping developers make the web" in a smaller, gray font. Below the logo, a subtext reads "Plunker is an online community for creating, collaborating on and sharing your web development ideas." There are two main buttons at the bottom left: a blue button labeled "Launch the Editor" and a green button labeled "Browse Plunks". To the right of these buttons are social media sharing icons for Twitter, Facebook, Google+, LinkedIn, and a plus sign.

Luego, aparece el editor y ya podes empezar a trabajar.

Poniéndole nombre a los archivos HTML

- No se pueden utilizar caracteres especiales como ñ, acentos, emojis o caracteres especiales
- No puedo dejar espacios entre palabras
- Sí puedo utilizar – (guion del medio) o _ (guion bajo)
- Lo mejor es trabajar con nombres cortos y referenciales (es decir que me den la idea de que está contenido en ese archivo)

Ejemplo de archivos mal nombrados:

- pagina de productos.html
- archivoEspañol.html
- ¿quiénes somos?.html

Ejemplo de archivos bien nombrados:

- productos.html
- archivo_espanol.html
- QuienesSomos.html

Del conjunto de archivos o páginas que van a conformar tu sitio Web, el primer archivo de tu sitio Web suele llamarse **index.html**. Cuando ingresamos a un sitio web tipeando su dirección no solemos identificar qué página dentro de ese sitio queremos cargar, por lo que el servidor donde ese sitio está alojado decide entregarnos lo que se llama Página Principal o Página de Inicio y esa página suele llamarse index (índice en inglés).

¿Qué es el Dominio?

Es un nombre que identifica a un sitio web en Internet y está directamente relacionado con algo llamado DNS (Servicio de Nombres de Dominio, por sus siglas en inglés por qué imaginate qué difícil sería tener que saber la dirección IP de un Sitio Web al que queremos visitar. Pero paremos un poco, ¿que es la dirección IP?

Es algo que identifica a tu dispositivo (teléfono celular, computadora, tablet, etc.) dentro de Internet, como si fuera su número de documento. Estas direcciones se parecen a 200.42.41.240 y

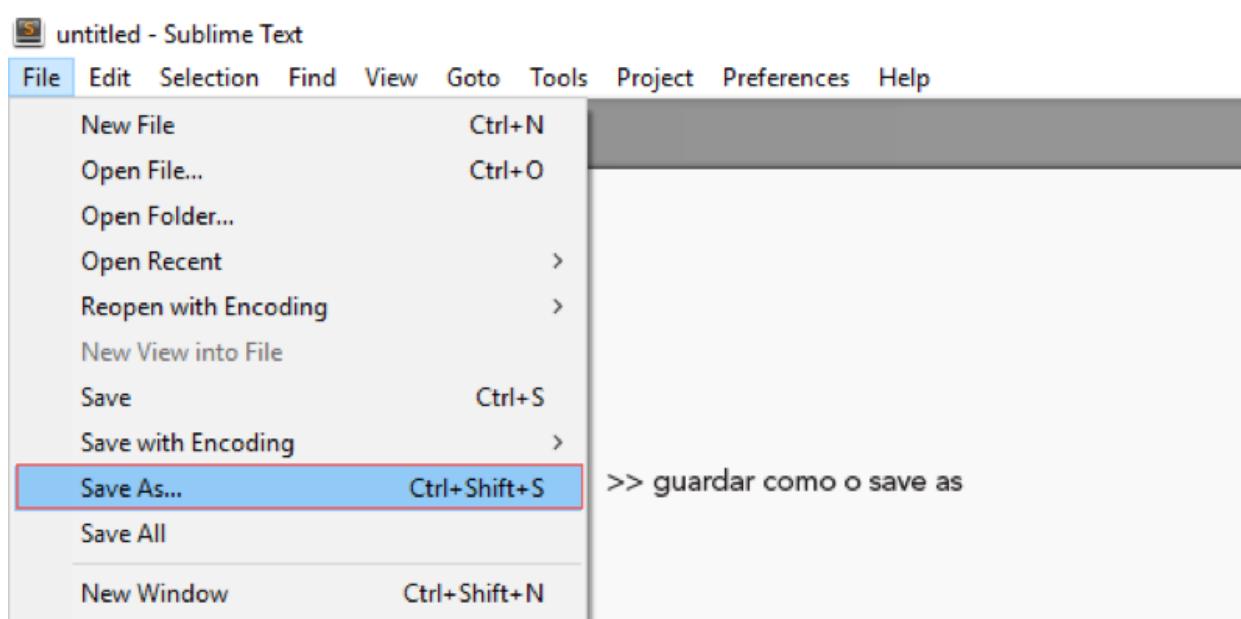
hay más de 4 mil millones de ellas. Entonces, si queremos ver una página web deberíamos tipar esa dirección en el navegador pero nadie las recuerda. Es mejor tipar www.argentina.gob.ar. Y eso es un dominio, es un sistema de nombres (terminados en .com, .com.ar, etc.) que se traducen a una dirección IP.

Creando nuestro primer archivo

Continuemos entonces creamos un archivo.html o si fue el primer archivo un index.html en nuestro editor de texto .

Ahora lo vamos a guardar. En general todos los editores funcionan de la misma forma; con acceder al manú de opciones y guardar el archivo con el nombre deseado será más que suficiente.

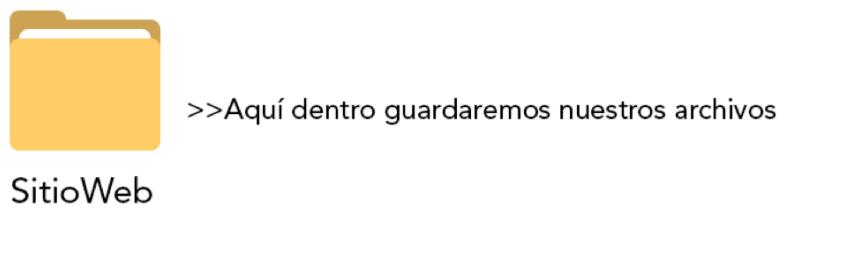
Ejemplo desde el Sublime Text usamos la opción Save As (Guardar Como).



¿Dónde lo guardo?

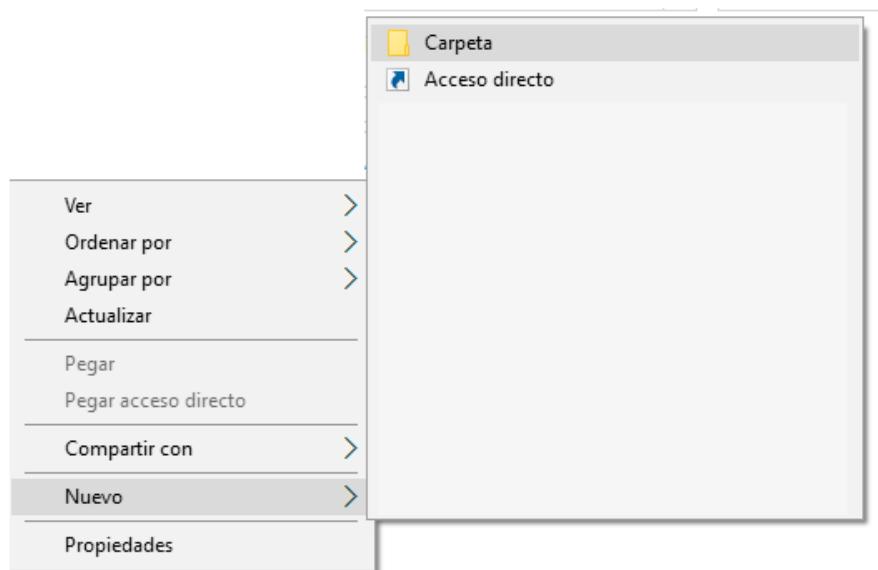
Por ahora lo vamos a guardar localmente, lo que significa que lo haremos en nuestra computadora. Te recomiendo siempre hacer primero una carpeta lo cual facilitará tu orden y evitará futuros errores.

Ejemplo:

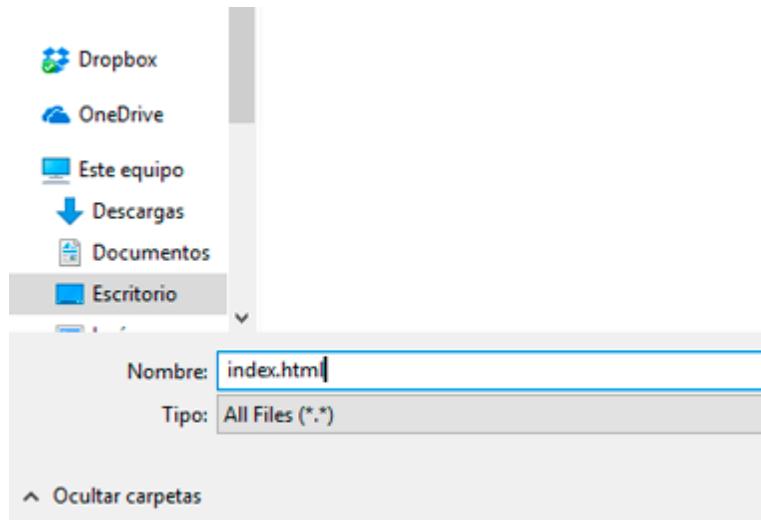


¡Ayuda! ¿Cómo creo una carpeta?

Como crear un carpeta en tu computadora es algo sencillo. Simplemente usás el botón derecho del mouse sobre el escritorio o Documentos y luego selecciono la opción nueva > carpeta, ejemplo:



Al momento de guardar el archivo simplemente seleccionó la carpeta deseada le hago doble clic y guardo dentro mi archivo.html, ejemplo:



Estructura básica de un archivo HTML

Ahora comenzaremos a escribir nuestra estructura. Trabajaremos con las primeras etiquetas, la estructura básica de un documento es la siguiente:

```
<!doctype html>
<title>Acá va el título de la página</title>
Acá va el contenido visible de la página
```

Ahora vamos a profundizar un poco más en cada una de estas líneas.

Declaración del Tipo de Documento

Le dice al navegador que nuestro archivo es un HTML, para ello se usa exactamente la siguiente línea en mayúsculas o minúsculas.

```
<!doctype html> o <!DOCTYPE HTML>
```

La cabecera del documento

La cabecera es un conjunto de etiquetas que dan información al navegador sobre nuestro archivo HTML pero no definen contenido que el usuario vea dentro del navegador. Lo más común de encontrar en esta sección es:

- **title** o título, Su definición es obligatoria, y debemos siempre incluirla, también tiene alto valor para que esta aparezca en los resultados de búsqueda si alguien quiere encontrar tu página. Es también el título que aparecerá en la pestaña del navegador cuando estemos usando este archivo HTML.
- Estilos de CSS que definen cómo se dibuja la página
- Archivos JavaScript que permiten agregar comportamiento interactivo a nuestra página
- Otra información relacionada con el posicionamiento web

De manera opcional, todo el contenido de la cabecera puede estar incluido en una etiqueta head (cabecera en inglés), por ejemplo:

```
<!doctype html>
<head>
    <title>Acá va el título de la página</title>
</head>
```

El cuerpo del documento

El cuerpo (body en inglés) contiene todo lo visible de nuestro documento, por ejemplo todos los elementos que veremos a continuación , elementos de texto, de imagen, de video, es decir todo aquello que se ve cuando entramos a una página web desde un navegador.

De manera opcional, podemos agrupar el cuerpo en una etiqueta body (cuerpo en inglés), siempre luego de la cabecera, como por ejemplo:

```
<body>
```

Acá va el contenido visible de la página
</body>

Anidación de etiquetas

Anidar es incluir una etiqueta dentro de otra, como si fuesen las mamuschkas (las famosas muñecas rusas que se colocan una dentro de la otra). Se le llama etiqueta padre a la que anida y etiqueta hijo a aquella que se encuentra anidada dentro.

Por ejemplo si usásemos este término diríamos que el title está anidado en el head.

```
<PADRE> <HIJO> </HIJO> </PADRE>
```

Primeros elementos de texto

Encabezados

Tenemos seis niveles de encabezados (o títulos visibles), todos completamente reutilizables, lo que significa que podemos usarlos la cantidad de veces que quieras en tu HTML. Los encabezados usan una etiqueta con una h y un número del 1 al 6.

Es necesario seguir un orden lógico de encabezados. el h1 es el encabezado más importante y el h6 es el encabezado menos importante. No es obligatorio usar todos los niveles de enunciados sólo aquellos que necesitemos. Por ejemplo, podemos tener un título, un subtítulo y otro nivel de enunciado dentro, lo que daría el uso de h1, h2, y h3.

```
<h1>encabezado de nivel 1 (el más importante)</h1>
<h2>encabezado de nivel 2<h2>
<h3>encabezado de nivel 3<h3>
<h4>encabezado de nivel 4<h4>
<h5>encabezado de nivel 5<h5>
<h6>encabezado de nivel 6<h6>
```

Párrafos de Texto

Los párrafos son los elementos que más utilizaremos en toda página, contienen la mayoría del contenido y al igual que los encabezados se situarán uno debajo del otro. ¿Por qué es así?, porque tanto los encabezados como los párrafos, son elementos de bloque, eso significa que de manera predeterminada ocupan todo el ancho disponible en la ventana del navegador y por tanto siempre el siguiente elemento se sitúa debajo.

```
<p>esto es un párrafo de texto</p>
<p>esto es otro párrafo de texto</p>
```

Atributos

Los atributos son información adicional sobre los elementos. Su sintaxis es la siguiente:

```
<elemento atributo="valor">
```

Si por ejemplo nos interesa agregarle al párrafo un nombre lo hacemos con el atributo conocido como id (de identificador), como por ejemplo:

```
<p id="descripcionProducto">esto es un párrafo con una descripción de producto</p>
```

Previsualizando nuestro trabajo

Siempre el HTML se ve dentro de un navegador. ¿Qué es un navegador? Es un programa que permite ver archivos.html. Hay muchos navegadores disponibles; los más conocidos son: Google Chrome, Mozilla Firefox, Opera, Safari y obviamente el que viene cuando tenemos el sistema Operativo Windows, que es Microsoft Edge o el anterior Internet Explorer.

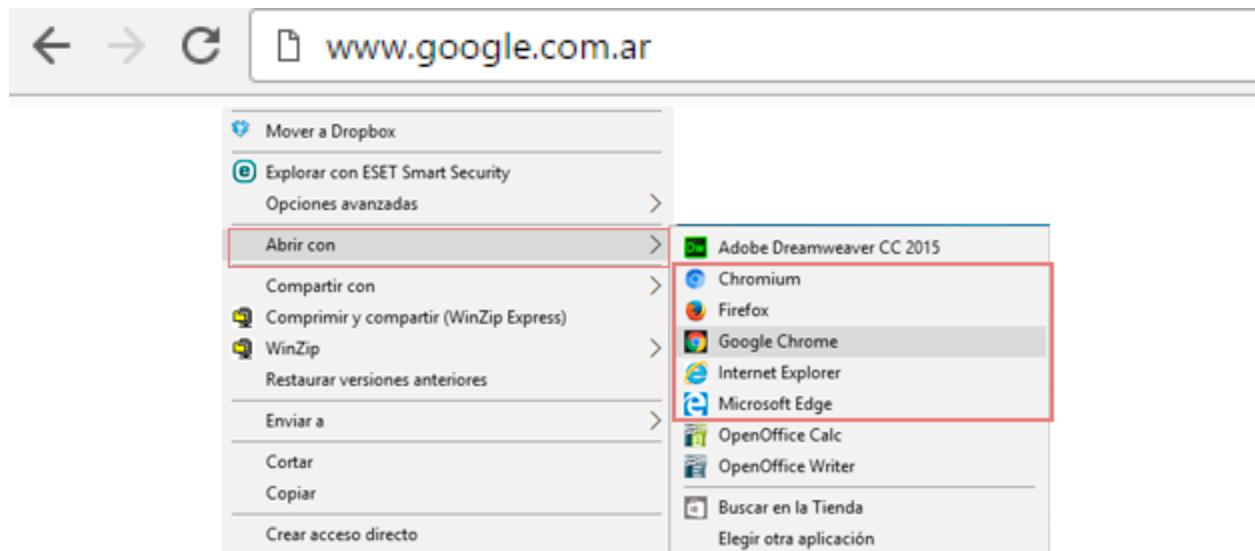
Ejemplo de iconos de navegadores:



Para abrir un archivo HTML en el navegador, simplemente hacemos botón derecho del mouse sobre el archivo y luego elegimos en “Abrir Con” y seleccionamos de la lista de navegadores instalados con el cual queremos ver el archivo.

Luego de abrirlo en alguno de estos navegadores también es posible copiar la dirección (la ruta o el lugar dónde está guardado) y pegar esa misma dirección en la barra de dirección de otro navegador que quiera.

Ejemplo de barra de dirección:



También podemos abrir el navegador manualmente desde su ícono y utilizar el menú Archivo -> Abrir y buscar el archivo HTML que queremos utilizar.

Otros Elementos

Comentarios del diseñador

Muchas veces queremos dejar en el código fuente HTML un mensaje para nosotros, para el diseñador/a y que no queremos que lo vea el usuario cuando navegue en nuestra página. Esto se llama un Comentario HTML y su sintaxis es

```
<!-- el comentario que queramos colocar aquí que puede ocupar más de una  
línea -->
```

Todo comentario va a ser ignorado por el navegador y el usuario no lo va a ver.

Saltos de Línea

Si bien cada vez que comenzamos un nuevo párrafo inicia automáticamente en una nueva línea (con un margen automático con el párrafo anterior), tenemos elementos que nos permiten hacer un salto de línea sin terminar el párrafo y son los elementos br y wbr. Estos elementos no llevan etiqueta de cierre, así que se usan como
 y en ese momento el navegador hará un salto de línea.

```
<p> Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod  
tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam,  
quis nostrud exercitation ullamco<br> laboris nisi ut aliquip ex ea commodo  
consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse  
cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non  
proident, sunt in culpa qui officia deserunt mollit anim id est laborum. </p>
```

La linea luego de br se verá debajo aún dentro del mismo párrafo. El <wbr> es parecido pero es un salto de línea opcional. Sólo si hiciera falta (porque es una palabra muy grande, por ejemplo) podemos especificar dónde sugerimos hacer un salto de línea al navegador, por ejemplo:

```
<p>Super<wbr>califragilístico<wbr>espialidoso</p>
```

Listas

Las listas son elementos que por definición están compuestas por varios ítems que pertenecen a un mismo grupo de información, como por ejemplo, una lista de países, una lista de artículos, un menú de un restaurant, etc. Existen tres tipos de listas:

- **Lista Ordenada** (etiqueta **ol**) Define elementos que llevan un orden natural, como los pasos de una receta (donde romper un huevo no puede ocurrir luego de batirlo) o un ranking de puntaje. Cada elemento de la lista ordenada debe ser especificado en una etiqueta **li** (ítem de lista). Se dibujan con números automáticamente (numerados desde el 1)
- **Lista Desordenada** (etiqueta **ul**) Define elementos que son parte de un grupo pero no llevan un orden natural, podría cambiarse el orden y siguen teniendo sentido, como por ejemplo una lista de alumnos en un curso. Cada elemento de esta lista también usa una etiqueta **li** (ítem de lista). Se dibujan con viñetas.
- **Lista de Definiciones** (etiqueta **dl**) Define elementos que tienen un título y una definición para ese título, como un diccionario. Por ejemplo, los datos personales de una persona donde se colocan los títulos en elementos **dt** y las definiciones en **dd**.

Veamos un ejemplo de listas en acción:

```
<h1>Provincias de Argentina</h1>

<ul id="listaDesordenada">
  <li>Salta</li>
  <li>Córdoba</li>
  <li>Buenos Aires</li>
</ul>

<h1>Ranking de Provincias por Población</h1>
<ol id="listaOrdenada">
  <li>Buenos Aires</li>
  <li>Ciudad Autónoma de Buenos Aires</li>
  <li>Córdoba</li>
</ol>

<h1>Datos de Río Negro</h1>
```

```

<dl id="listaDefiniciones">
    <dt>Nombre Completo</dt>
    <dd>Provincia de Río Negro</dd>
    <dt>Capital</dt>
    <dd>Viedma</dd>
    <dt>Población</dt>
    <dd>52 754</dd>
</dl>

```

Y así se ve en el navegador

Provincias de Argentina

- Salta
- Córdoba
- Buenos Aires

Ranking de Provincias por Población

1. Buenos Aires
2. Ciudad Autónoma de Buenos Aires
3. Córdoba

Datos de Río Negro

Nombre Completo	
Provincia de Río Negro	
Capital	
Viedma	
Población	
52 754	

Mediante CSS, luego, vamos a tener la posibilidad de modificar el estilo visual y gráfico de todos los elementos en pantalla si no nos gusta el estilo que pone el navegador.

Cambiar tipo de viñetas

Si queremos cambiar el tipo de estilo de viñetas de números a letras por ejemplo, o elegir otras viñetas dentro de las listas desordenadas lo haríamos con el atributo **type** que se puede aplicar al elemento de la lista (por ejemplo ul o ol) o a cada ítem (li), como en el siguiente ejemplo que vemos desde el editor:

```

<ul>
<li type="square"> item con cuadrado relleno </li>
<li type="circle">item con circulo vacio </li>
<li>item que tiene de manera predeterminada disc,
circulo relleno</li>
</ul> <!--ejemplo de lista desordenada -->

<ol>
<li> item que de manera predeterminada tiene 1 es decir decimal </li>
<li type="a"> item con alfabeto latino en minúscula </li>
<li type="A">item con alfabeto latino en mayúscula</li>
<li type="i">item con letras romanas en minúscula</li>
<li type="I">item con letras romanas en mayúscula </li>
</ol>

```

Si el ejemplo previo lo ejecutamos en el navegador se vería de la siguiente forma:

- item con cuadrado relleno
 - item con circulo vacio
 - item que tiene de manera predeterminada disc, circulo relleno
- 1. item que de manera predeterminada tiene 1 es decir decimal
 - b. item con alfabeto latino en minúscula
 - C. item con alfabeto latino en mayúscula
 - iv. item con letras romanas en minúscula
 - V. item con letras romanas en mayúscula

Elementos de formato de texto

Existen algunos otros elementos que son sumamente interesantes por ejemplo

```

<p>
    <strong>texto muy importante dentro del contenido del párrafo.</strong>
    <em>texto que queremos enfatizar dentro del contenido de mi página.</em>
    <b>texto que aparecerá en negrita en el navegador.</b>
    <i>texto que aparecerá en itálica en el navegador.</i>
</p>

```

Si viésemos lo anterior desde el navegador se vería de la siguiente forma:

texto muy importante dentro del contenido del párrafo *texto que queremos enfatizar dentro del contenido de mi página* **texto que aparecerá en negrita en el navegador** *texto que aparecerá en itálica en el navegador*

Imágenes

Las imágenes y los elementos gráficos son un elemento muy importante al momento de mostrarle algo al usuario que diga quienes somos o que estamos comunicando con nuestra página web.

Veamos como incorporar una imagen usando la etiqueta **img** que no requiere cierre:

```

```

El atributo src (origen)

Src es la abreviatura de *source* en inglés que significa origen. Me permite indicar la ruta o lugar donde está guardado el archivo de imagen con el cuál voy a trabajar. Puede estar trabajada de manera absoluta o relativa, por ejemplo:

El atributo alt (texto alternativo)

Cumple el propósito de describir la imagen en texto que sirve para que se vea como ayuda si la imagen tarda en aparecer, para que los buscadores entiendan de qué se trata la imagen y para ayudar a personas con discapacidades visuales a entender qué se encuentra en esa sección.

El atributo width (ancho)

Me permite indicar el ancho de las mismas, podemos trabajar con un numero entero que representa su medida en pixeles o un porcentaje que indica su medida en relación a su elemento contenedor, por tanto si le coloco 50% será la mitad de ancho de su elemento contenedor, como un párrafo o la página misma

El atributo height (alto)

Me permite indicar el alto de las mismas, podemos trabajar con un numero entero que representa su medida en pixeles o un porcentaje que indica su medida en relación a su elemento contenedor.

Las imágenes pueden manejar diferentes formatos, y también puede ser que alguien haya diseñado esa imagen o que sea producto de una fotografía de tu celular o tu cámara.

Vínculos o *Links*

Un vínculo o link es una zona de la página web que al presionarla con el mouse o con el dedo en un teléfono o Tablet nos llevará a otro archivo HTML, ya sea dentro del mismo sitio o fuera del sitio. Para ello se utiliza el elemento con la letra `<a>` de ancla.

Vamos un ejemplo de su uso:

```
<!DOCTYPE html>
<html>
<head>
    <title>mi primer sitio web</title>

</head>
<body>
<a href="destino del vinculo" title="descripcion del vinculo" target="nos indica donde abrirá el destino del vinculo"> texto de anclaje </a>

</body>
</html>
```

Vemos que entre `<a>` y `` colocamos el contenido que queremos que actúe como contenido de anclaje, o sea la zona que podemos cliquear o tocar. Puede ser un texto, una imagen o contenido más complejo.

Atributos de los vínculos

Este elemento como se ve en el ejemplo anterior, consta de tres atributos a saber:

El atributo href o dirección

Este atributo indica hacia donde nos lleva el vínculo, su valor puede tener dos variantes, por caso:

- **Rutas relativas:** Son aquellas dentro de nuestro propio servidor, por esa razón sólo es necesario indicar la carpeta o dirección del archivo hacia donde nos vamos a dirigir y nada más.

```
<a href="destino.html">Texto del Vínculo</a>
```

- **Rutas absolutas:** Tienen un destino que nos lleva por fuera de nuestro servidor, como puede ser otra página web y comienza con http:// o https://

```
<a href='http://www.itmaster.com.ar">ITMaster Academy</a>
```

- **Acciones:** Tienen como destino no una página web, sino abrir otra aplicación, como puede ser la aplicación para enviar un email, hacer una llamada telefónica o abrir Whatsapp.

```
<a href="tel:+541150320077">Llamanos</a>
```

El atributo target o ventana de destino

Si probamos los anteriores vínculos cuando hacemos clic en ellos, la página destino se abre en la misma pestaña del navegador, para poder cambiar eso debemos trabajar con el atributo target, o ventana de destino.

Se suele trabajar con dos valores para este atributo:

- **_self:** Este valor es el valor predeterminado, y me indica justamente que el valor del href se abrirá en la misma ventana o pestaña
- **_blank:** Este valor indica que el valor del href se abrirá en una pestaña o ventana nueva del navegador

El atributo title o título

El último atributo es el title, este describe al vínculo es obligatorio por cuestiones de accesibilidad para personas con capacidades diferentes y también para describir el vínculo para los buscadores que vean nuestra página web. También si pasamos el mouse por encima el vínculo notaremos que el title genera un mensaje de ayuda o etiqueta amarilla con la descripción

Si bien el title es un atributo sólo obligatorio en los vínculos, también se puede aplicar a cualquier elemento de nuestro HTML con el fin de describirlo o de generar la ventana de ayuda

Veamos algunos links de ejemplo

```
<!DOCTYPE html>
<html>
<head>
    <title>mi primer sitio web</title>

</head>
<body>
<a href="archivo.html" title="vinculo al archivo" target="_blank"> vinculo relativo
</a>
<a href="http://www.otrapagina.com/archivo.html" title="vinculo al archivo"
target="_blank"> vinculo relativo </a>

</body>
</html>
```

Vínculo de Correo Electrónico

El vínculo de correo electrónico sólo tiene de diferente al vínculo anterior que el valor del href comienza con la palabra **mailto:** para luego contener el correo electrónico que aparecerá en el para o to del correo electrónico predeterminado del usuario, eso depende enteramente de él y puede ser desde Outlook hasta Gmail o Yahoo.

Ejemplo desde el editor:

```
<!DOCTYPE html>
<html>
<head>
    <title>mi primer sitio web</title>

</head>
<body>
<a href="mailto:info@mimail.com" title="Vinculo de correo electrónico"> Vinculo de
correo electrónico </a>

</body>
</html>
```

Si vemos este vínculo en el navegador y le hacemos clic y por caso tenemos configurado Gmail como correo electrónico predeterminado veremos que sucede lo siguiente:

Mensaje nuevo

info@mimail.com

Asunto

Anclas Internas

El último tipo de vínculo que veremos son las anclas, éstas permiten navegar a un punto específico de una página HTML que suele ser muy larga. Entonces podemos generar vínculos o menús internos que nos permitan movernos en todo el documento.

Las anclas tienen dos partes, una es el vínculo en sí, es decir como de costumbre aquello sobre lo que hace clic el usuario, con la salvedad que luego del href, trabajaremos con un carácter # y el identificador del elemento a donde nos llevará esta ancla dentro del documento html.

Ejemplo desde el editor:

```
<a href="#menu">Ir al Menú</a>
<!-- Aquí puede haber mucho código HTML --&gt;
&lt;ul id="menu"&gt;
    &lt;li&gt;Opción 1&lt;/li&gt;
    &lt;li&gt;Opción 2&lt;/li&gt;
&lt;/ul&gt;</pre>
```

En este caso al hacer click en "Ir al Menú" el navegador desplazará la barra de desplazamiento (scroll) hasta el elemento cuyo identificador (atributo id) sea exactamente menú. Notemos que cuando hacemos el vínculo usamos el carácter # de prefijo y no lo usamos cuando definimos el identificador.

El lenguaje CSS

CSS es un lenguaje como lo es HTML; significa Cascading StyleSheet (Hoja de Estilos en Cascada). A diferencia de HTML, este lenguaje no define el contenido de la página sino el cómo se ven esos elementos de HTML. Es decir en HTML indicamos que por ejemplo un `<p> </p>` es un párrafo o que un `<h2> </h2>` es un encabezado de nivel 2 , pero no indicamos ni su color, ni su tipografía ni otras características que no tengan que ver con el dato en sí mismo y qué significa. Es importante separar lo que algo significa de cómo queremos que se vea. Para HTML no importa el cómo se vea sino lo que signifique. Para CSS importa cómo vamos a mostrar esos elementos en pantalla.

Para trabajar con CSS tenemos tres formas de hacerlo:

Archivo de Estilos Externo



La forma más recomendable de hacerlo es a través de un archivo externo. ¿por qué es la más recomendable?, porque simplemente nos permite modificar el diseño de un Sitio Web Entero con todos sus HTML con tan sólo un sólo archivo CSS. Lo hacemos a través de la etiqueta o elemento link, que contiene dos atributos fundamentales para que esto funcione:

- **href** o ruta del archivo: Me indica donde está guardado el archivo.css que va a afectar a este archivo html
- **rel** o relación: Establece qué relación tiene ese archivo con el html y en este caso como es nuestra "hoja de estilos" (*stylesheet* en inglés), ese es el valor que lleva rel="stylesheet"

Si quisiéramos que nuestro archivo HTML estuviese relacionado con un CSS, lo que haríamos sería colocar una etiqueta `<link>` usualmente en la sección cabecera de nuestro HTML (arriba de todo, antes del contenido visible).

```
<!doctype html>
<title>Mi Primer Sitio Web</title>
<link href="css/estilos.css" rel="stylesheet">
```

Lista de estilos Interna

Esta forma de trabajo define los estilos que vamos a usar no en un archivo externo, sino dentro del mismo HTML (o sea, tendríamos dos lenguajes en el mismo archivo, HTML y CSS). Esta modalidad de implementar CSS en HTML hace que a diferencia del modelo anterior, el único HTML afectado por el CSS sea donde estamos escribiendo y ningún otro del sitio web (en caso de tener muchos archivos HTML).

Para definir estilos internamente se utiliza una etiqueta **style**.

Ejemplo:

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <title>mi primer sitio web</title>
5 <style>
6 /*aca escribo css*/
7
8
9 </style>
10 </head>
11 <body>
```

Aquello que está escrito entre `/* */` es un comentario en CSS, los comentarios permiten o transmitir información para nosotros mismos o para otros desarrolladores sin afectar lo que está escrito en el lenguaje.

Estilos en línea

La metodología conocida como estilos en línea, se hace a través del atributo **style** sobre el elemento al cual queremos aplicarle el estilo. Como es un atributo, sólo afecta al elemento al cual le aplico CSS y no puede ser reusado. Por ejemplo si tenemos 100 párrafos y queremos que sean de color gris debemos escribir el color manualmente en cada uno de los párrafos por lo cuál es la forma menos recomendada

Ejemplo :

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4  |   <title>mi primer sitio web</title>
5
6  </head>
7  <body style="propiedad:valor;">
8
9  </body>
0  </html>
```

Entendiendo las Reglas de Estilos

CSS trabaja con reglas de estilo las cuales se estructuran de la siguiente forma:

```
selector {  
    propiedad: valor;  
    propiedad: valor;  
}
```

Ahora vemos qué significa cada cosa que vemos aquí

El Selector

Es una forma de elegir a qué elementos quiero aplicar un estilo, o de pedirle al navegador que busque elementos que cumplan con ciertos requisitos. Por ejemplo, podríamos pedirle que ponga en color azul a todos los párrafos ("todos los párrafos" sería el selector) o que use un borde rojo en aquellos campos de texto de un formulario que sean obligatorios ("campos de texto que sean obligatorios" sería el selector).

Existen diferentes tipos de selectores. Ahora vamos a ver los más comunes.

Selector de Elemento

Este tipo de selector está formado a partir del nombre del elemento al cual queremos aplicar un estilo, por ejemplo si quisiera afectar a los párrafos lo haría de la siguiente forma

```
p {  
    propiedad: valor;  
}
```

La característica de este selector reside en que aplica un estilo a todos los elementos de un tipo (a todos los párrafos) y muchas veces es poco útil dado que nuestro HTML podría contener muchos de ese tipo y podríamos no querer que todos estén con el mismo estilo.

Selector único de elemento por identificación

Este selector apunta a un solo elemento en el HTML a partir de su identificador (atributo id) el cual sigue las mismas reglas de nomenclatura que los nombres de archivos HTML, sumado a que no debe nunca comenzar con un número, por ejemplo. En este caso el selector lleva un carácter # de prefijo y luego el id al que queremos apuntar, por ejemplo si tenemos una imagen cuyo identificador es logo-empresa (como en) podemos usar el siguiente selector para definir un estilo:

```
#logo-empresa {  
    propiedad: valor;  
}
```

Dentro de un mismo documento HTML no puede haber dos elementos con el mismo identificador, por lo que estamos hablando de una referencia única.

Selector de Clase

Si quisiera afectar a varios elementos (ni todos de un tipo ni uno solo), los selectores anteriores no serían suficientes, para esto puedo trabajar con las clases.

Una clase es una forma que nos da HTML de clasificar elementos por tipo. La clasificación es cualquier nombre que queramos pero que siga las reglas de nombres ya vistas (por ejemplo, sin espacios). Por ejemplo, podríamos tener tres párrafos y clasificar dos de ellos como importantes con el atributo class:

```
<p class="importante">Este es un párrafo muy importante</p>  
<p>Este párrafo no parece importante</p>  
<p class="importante">Pero este último es importante también</p>
```

Para aplicar estilos CSS a los elementos de clase "importante" debemos usar un selector que lleva un prefijo de un punto (sin espacios), como por ejemplo:

```
.importante {  
    propiedad: valor;
```

{}

Si bien en el ejemplo anterior se trabajó sobre el mismo tipo de elemento es decir párrafos, puedo aplicarle la misma clase a cualquier tipo de elemento dado que no debe ser el mismo

```
<h1 class="importante">El título también es importante</h1>
<p class="importante">Este es un párrafo muy importante</p>
<p>Este párrafo no parece importante</p>
<p class="importante">Pero este último es importante también</p>
```

Propiedades y Estilos en CSS

Ya sabemos cómo elegir a qué elementos vamos a aplicar estilos CSS con los selectores pero no sabemos todavía cómo definir esos estilos. Es momento de hablar de las propiedades de estilos en el lenguaje CSS que es lo que va dentro de las llaves de una regla.

Vamos a empezar hablando de la tipografía, o fuente, la cual define información sobre el tipo de letra que vamos a ver en pantalla cuando se ejecute el archivo HTML en un navegador. Veamos algunas propiedades.

Familia de Tipografía: font-family

Esta propiedad hace referencia al tipo de tipografía que llevará nuestro texto. Tenemos una cantidad infinitas de tipografías que se puede utilizar, por ejemplo Arial, Verdana, Times New Roman, etc.

```
p { font-family: Arial }
```

En el ejemplo anterior, todos los párrafos estarán en Arial, si quiero que por ejemplo lo hagan los encabezados haremos lo siguiente:

```
h1 { font-family: Arial }
```

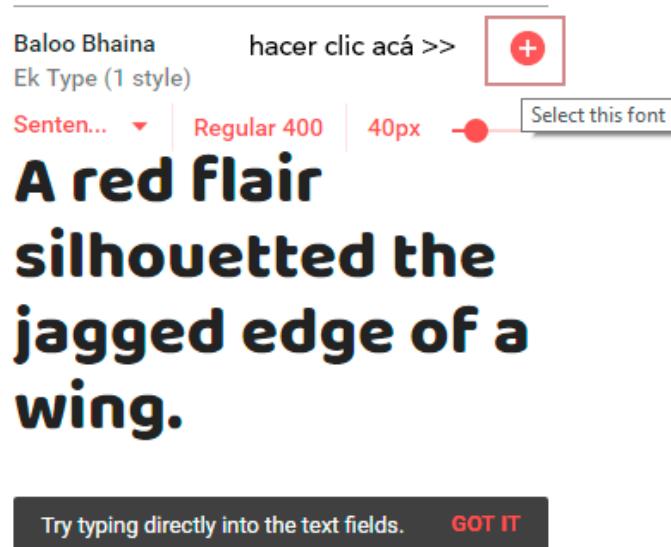
Si necesitase que por ejemplo, en caso de que el navegador no encuentre una tipografía en la computadora del usuario, busque otras alternativas, siempre entendiendo que la primera es la alternativa más elegida, optaré por crear un camino de fuentes siguiente lo siguiente lógica:

```
h1 { font-family: sugerencia1,sugerencia2,sugerencia2,tipoGenerico}
```

Puedo sugerir cuantas tipografías quiera, incluso , puedo trabajar con una fuente que yo desee por ejemplo desde Google Fonts.

Google Fonts es un servicio gratuito que nos ofrece tipografías para usar en nuestra Web. Hay que ingresar en <https://fonts.google.com/> en un navegador y luego elegir la tipografía con la que queremos trabajar.

Al elegir la tipografía con la cual quiero trabajar, simplemente le hago clic al ícono de más que encuentro allí, ejemplo



Luego la idea es utilizar el tag que Google nos ofrece ahí mismo, por ejemplo:

STANDARD @IMPORT

```
<link href="https://fonts.googleapis.com/css?family=Baloo+Bhaina" rel="stylesheet">
```

Specify in CSS

Use the following CSS rules to specify these families:

```
font-family: 'Baloo Bhaina', cursive;
```

Si notamos, debajo nos indica cómo debemos llamar a esa tipografía en nuestra propiedad, ejemplo desde nuestro propio editor de texto

```
<!DOCTYPE html>
<html>
<head>
    <title>mi primer sitio web</title>
    <link href="css/estilos.css" rel="stylesheet">
    <link href="https://fonts.googleapis.com/css?family=Baloo+Bhaina" rel="stylesheet">

</head>
```

Entonces en nuestro CSS luego de incluir el link que incorporará la tipografía podemos usar:

En el caso anterior, he utilizado body para aplicar mi tipografía, esto es porque aprovecho la
`body {font-family: 'Baloo Bhaina', cursive;}`

interesante característica, que los elementos hijos (anidados, que se encuentran dentro de otros) heredan propiedades de tipografía de los elementos padres (que anidan, o que los contienen), en este caso todo mi documento tendrá esta tipografía pues el body (cuerpo de la página) anida a todos los elementos visibles.

```
p {font-weight: bold;}
h1 { font-weight: normal}
```

Negritas: font-weight

Esta propiedad hace referencia al peso de fuente de un elemento de texto, si bien existen una variada cantidad de valores en general trabajaremos con **bold** (negrita) y **normal**, ejemplo

En el ejemplo anterior, hemos transformado los párrafos en negrita, y a los encabezados en normal, esto es porque los encabezados así como el elemento **strong** , **b** y demás elementos de manera predeterminada están en negrita y hemos elegido quitarles esa característica y dejarlos en normal.

Estilos de letra: font-style

Esta propiedad permite transformar o no un elemento de texto en itálica, ejemplo de uso

```
p {font-style: italic;}  
.varios em { font-style: normal}
```

En el ejemplo anterior se ha utilizado un tipo de selector nuevo: se llama selector de descendientes, eso indica que yo afecto solamente a los em (la etiqueta em) que se encuentren dentro de algún elemento cuya clase sea varios ejemplo desde el código HTML

```
<p class="varios"> Parrafo con un <em> texto destacado </em></p>  
<h2 class="varios"> enunciado con un<em> texto destacado</em> </h1>
```

Los em anteriores se encuentran ambos anidados en elementos que tienen la clase varios por tanto ambos em perderán su característica predeterminada de estar en itálica y pasarán a verse en normal es decir en este caso sin itálica

Tamaño de Letra: font-size

Esta propiedad es sumamente interesante y hace referencia al tamaño de texto del selector en cuestión. Tenemos varias posibles medidas para trabajar aquí.

Hay muchas formas de definir el tamaño de la tipografía pero veamos las más usuales.

- con constantes que pueden ser (de más chico a más grande): x-small, smaller, small, medium, large, larger, x-large, xx-large. Por ejemplo: **font-size: x-large;**
- usando medida de pixel (px) que depende del medio donde veamos la página, por ejemplo: **font-size: 14px** daría una tipografía de 14 píxeles. Notemos que no se coloca espacio entre el número (14) y la unidad de medida (px)
- usando em, relativo al tamaño medio de ese elemento en la tipografía actual. Por ejemplo: 0.5em significa la mitad del tamaño medio y 2em el doble. Ejemplo: **font-size: 1.4em**

Color de frente: color

Esta propiedad es sumamente interesante dado que nos permite cambiar el color de texto de cualquier elemento. Puedo trabajar con diferentes valores posibles:

- nombres de colores en inglés: como red, green, blue, yellow
- cantidad de rojo, verde y azul
- valores hexadecimales

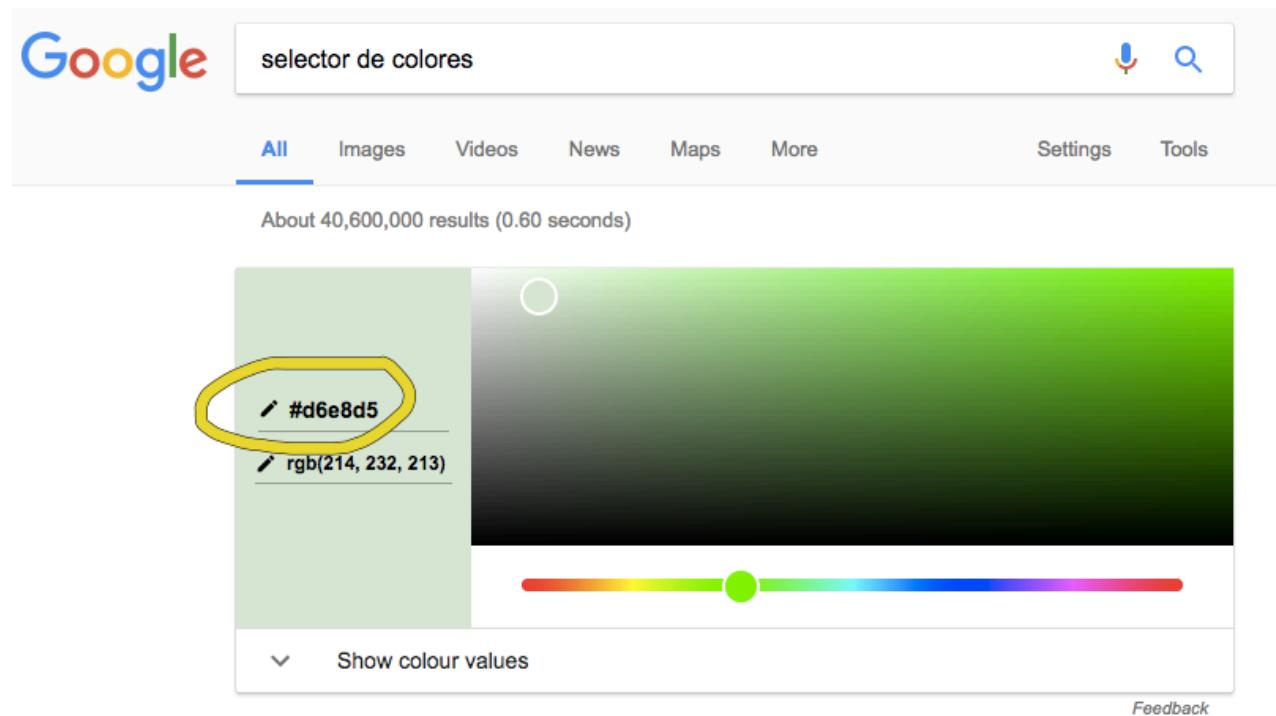
Lo interesante es saber de dónde saco que color quiero representar en mi web. Bueno hay muchas herramientas online, por ejemplo ingresando a <http://html-color-codes.info>.

En principio esta página es mucho más porque si notan tenemos opciones tales, como saber los nombres de los colores, paletas utilizadas, y demás pero si por caso queremos saber el color de nuestro diseño la opción es:

<http://html-color-codes.info/colors-from-image/>

Simplemente hay que subir la imagen , presionar el botón que dice "show image" y con el cursor de mouse presionar sobre el color que queremos trabajar, debajo aparecerá el valor que lo representa, ¡listo! Ya lo podes usar

También podés buscar en Google "selector de colores" y encontrás un lindo selector que te dará el valor a usar dentro de CSS:



Te doy un ejemplo de cómo usarlo desde CSS:

```
p {font-style: italic; color: lightBlue;}
h1 { color: red;}
em { color: #f0f;}
strong { color: #ff0000;}
i { font-style: normal; font-weight: bold; color: rgb(0,255,0)}
```

En mi HTML se verá lo siguiente:

```
<body>
<p> Parrafo con un <em> texto destacado </em></p>

</body>
```

Transformación de Texto: text-transform

Es una propiedad que me permite transformar las mayúsculas en minúsculas y viceversa, por

```
p { text-transform: uppercase;} /*transformo el texto en mayúsculas*/
p { text-transform: lowercase;} /*transformo el texto en minúsculas */
p { text- transform: capitalize;} /*transformo el texto donde la primer letra de cada
palabra está en mayúsculas*/
```

ejemplo

Alineación de Texto: text-align

Esta propiedad indica hacia donde está alineado el elemento texto. Ejemplo:

```
p { text-align: center;} /*alineado hacia el centro en referencia a su tamaño*/
p { text-align: left} /*este es el valor predeterminado es decir hacia la izquierda */
p { text- align: right;} /*alineado hacia la derecha*/
p { text- align: justify;} /*justificado, todas las lineas menos la última tendrán
el mismo largo*/
```

Contenedores de Contenido

Los contenedores son elementos que no tienen visualización gráfica pero nos permiten construir nuestra Web de forma más prolija agrupando contenido al cual podemos por ejemplo luego aplicarle CSS:

Encabezados Visibles

Muchas veces una página web tiene en su parte visible un encabezado en la parte superior. Suele contener un logo, un menú de navegación u otro contenido informativo sobre el contenido en cuestión. Para ello podemos usar el elemento contenedor `<header>` dentro del cual colocaremos todo el contenido visible que queremos que sea parte del encabezado

```
<header>
<h1> Bienvenidos a nuestro Sitio Web </h1>
<ul>
<li>
<a href="index.html"> Inicio </a> </li>
<li> <a href="servicios.html"> Servicios </a> </li>
<li> <a href="contacto.html"> Contacto </a> </li>
</ul>

</header>
```

Aquí hemos anidado los vínculos en una lista desordenada, esta es la mejor forma de estructurar un grupo de links del mismo tipo.

Zona de Navegación

Es un elemento que debe contener los links o vínculos principales que hacen a la navegación del sitio. Para esto se usa el contenedor <nav>.

```
<header>
<h1> Bienvenidos a nuestro Sitio Web </h1>
<nav>
<ul>
<li>
<a href="index.html"> Inicio </a> </li>
<li> <a href="servicios.html"> Servicios </a> </li>
<li> <a href="contacto.html"> Contacto </a> </li>
</ul> </nav>
```

Pie de Página

El pie de la página suele contener información de contacto, autor o copyright y muchas veces suele repetir links o vínculos ubicados en otras partes de nuestra página. Para ello se utiliza la etiqueta <footer>

```
<footer>

<p> copyright 2017 todos los derechos reservados </p>

<p> Contáctenos <a href="mailto:info@miweb.com" title="correo electrónico">info@miweb.com </a> </p>

</footer>
```

Sección

Es una parte del todo, es decir un elemento que no puede comprenderse por fuera del todo. Por ejemplo, un artículo de un diario puede estar dividido en varias secciones pero cada sección sola está incompleta o no es útil. Una página de detalle de un producto a vender puede tener varias secciones, como el precio, las fotos, la ficha técnica. Cada sección es parte de un todo pero no sirve por sí sola.

Para esto se utiliza el elemento `<section>`, el cual puede tener otros contenedores anidados como ser otros `<section>` (subsecciones), y sus propios `<header>` y `<footer>`

```
<section>
<header>
<h2> Servicios </h2>
<ol>
<li> <a href="#uno"> Servicio 1 </a> </li>
<li> <a href="#dos"> Servicio 2 </a> </li>
<li> <a href="#tres"> Servicio 3 </a> </li>
</ol>

</header>

<h3> Servicio 1 </h3>
<p><a id="uno"> </a> Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod
tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam </p>

<h3> Servicio 2 </h3>
<p><a id="dos"> </a> Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod
tempor </p>
<h3> Servicio 3 </h3>
<p><a id="tres"> </a> Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod
tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, </p>
</section>
```

Artículos o Items

Un artículo representa una unidad de información que si bien está contenida en el sitio web, puede ser independiente por ejemplo podemos redistribuirlo en una red social y comprenderlo por sí

```

<article>
<header>
<h2> Articulo Especial </h2>
<p> Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod
tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam,
quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo
consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse
cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non
proident, sunt in culpa qui officia deserunt mollit anim id est laborum. </p>
</header>

<p> Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod
tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam,
quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo
consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse
cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non
proident, sunt in culpa qui officia deserunt mollit anim id est laborum. </p>

<footer>
<p> Autor: Pedro Perez </p>
<p> Fecha: 2017 </p>
<p> Lugar: Argentina </p>
<p> Mail: <a href="mailto:info@mimail.com"> info@mimail.com </a> </p>
</footer>
</article>

```

mismo. En un sitio de venta online, cada artículo que vendemos puede ser un `<article>` y en un diario cada nota puede ser un `<article>`. Usualmente si la información de nuestro HTML surge de una base de datos, cada registro en una tabla es candidato a ser un artículo de HTML. Como con secciones, un artículo puede estar dividido en secciones, encabezados y pies en base a cuán compleja sea la información contenida.

Contenedor Principal

Muchas veces cuando tenemos un contenedor de encabezado (header) y uno de pie de página (footer), nos queda el contenido del medio que no sabemos qué contenedor usar. Para eso existe el contenedor principal (`<main>`) que podemos utilizar opcionalmente.

Divisor Genérico

Cuando queremos agrupar un conjunto de elementos HTML bajo un contenedor pero no nos parece que todos los contenedores que vimos hasta ahora parezcan adecuados podemos usar un divisor genérico, que divide partes del documento HTML sin dar ningún significado de por qué. Para esto usamos la etiqueta <div>.

Trabajando con bloques en CSS

Ahora que ya conocemos muchas etiquetas HTML, tanto de texto, links como de contenedores, sigamos descubriendo propiedades de CSS que podemos aplicar en nuestros diseños.

Propiedades de Caja

La caja es la zona rectangular que contiene a un elemento de tipo bloque, que son aquellos que contienen texto o contenido, como ser un párrafo, un encabezado de tipo h1-h6 o un contenedor, como section o header.

La caja está compuesta desde adentro hacia afuera por:

- El contenido propiamente dicho con una dimensión
- Espacio de relleno
- Bordes
- Márgenes

Dimensiones de la caja: width y height

El ancho (width) hace referencia al ancho de un elemento en pixeles o con porcentaje. Si usamos porcentaje siempre depende del elemento padre es decir que si digo que un elemento tiene 50% quiere decir que es la mitad de su elemento contenedor.

El alto (height) hace referencia al tamaño de alto de un elemento. Posee los mismos valores y características que la propiedad width.

Veamos algunos ejemplos de su uso:

```

div { width: 100%; height: 200px; }
/*este elemento ocupa el 100% en referencia a su contenedor , y
de alto tiene 200px*/

header { width: 900px; }
/*este elemento no tiene alto, ya que si no indico el mismo, fluye según sea la cantidad
de contenido que tenga, tiene un ancho de 900px*/
footer {width: 100%; height: 200px}
#pie { height: 400px;}
```

En el caso anterior, vemos como al final hemos escrito por un lado footer y por el otro lado un id, la explicación sería la siguiente desde el archivo.html vinculado con este archivo.css

```

<!DOCTYPE html>
<html>
<head>
    <title>mi primer sitio web</title>
    <link href="css/estilos.css" rel="stylesheet">

</head>
<body>
<footer id="pie"> footer con el id pie, si bien toma las características
del footer en general, también toma específicamente sus propias características, en
este caso tiene un width de 100% pero tiene un height de 400px </footer>

<footer>
footer sin id, es decir tomas las características del selector footer </footer>

</body>
</html>
```

Márgenes

Los márgenes nos permiten establecer cuál es el espacio entre elementos contiguos (arriba, abajo, derecha e izquierda) o con los bordes de la página si no hubiera más elementos. Se pueden establecer valores de márgenes en pixeles, porcentajes y todas las medidas ya vistas, y se pueden establecer de diversos modos, como todos los márgenes iguales o valores separados por cada lado.

A continuación veremos varios ejemplos de uso:

```
p{ margin-top: 10px; margin-left: 2px; margin-bottom: 12px; margin-right: 0; }
/*en el caso anterior, al párrafo le hemos puesto
margen de arriba > 10px
margen de abajo > 12px
margen de izquierda > 2px
margen de derecha > 0 -> cuando ponemos 0 si se quiere se puede omitir la medida de longitud */

p { margin: 2px 3px; }

/*esto anterior se llama shorthand es para escribir menos en nuestros estilos, y significa que el
margen de arriba y abajo será de 2px y el margen de izquierda y derecha será de 3px*/

p {margin: 1px 4px 2px;}

/*En el caso anterior, estamos diciendo que el margen de arriba será de 1px, el de izquierda y
derecha será de 4px y el de abajo será de 2px*/

p { margin: 3px 5px 13px -2px; }
/*En el caso previo, debemos seguir la orientación de las agujas del reloj, es decir
arriba 3px, derecha 5px, abajo 13px , izquierda -2px - nótese que se puede trabajar con valores
negativos */
```

Para que un elemento se sitúe centrado horizontalmente en referencia a su contenedor, necesitamos trabajar con la propiedad `margin-left` y `margin-right` pero con el valor especial **auto** (de automático). La premisa para que este funcione es que tenga un `width` explicitado ejemplo:

```
div { width: 100px; height: 20px; margin: 0 auto; height: 200px; }
```

En este caso indicamos que los márgenes superior e inferior sean cero (0 no requiere de unidad como px o %) y que los márgenes izquierdo y se calculen automáticamente (auto).

```
p{ padding-top: 10px; padding-left: 2px; padding-bottom: 12px; padding-right: 0; }
/*en el caso anterior, al párrafo le hemos puesto
padding de arriba > 10px
padding de abajo > 12px
padding de izquierda > 2px
padding de derecha > 0 */

p { padding: 2px 3px; }

/*Significa que el padding de arriba y abajo será de 2px y el padding de izquierda y derecha será
de 3px*/

p { padding: 3px 5px 13px -2px; }
/*En el caso previo, debemos seguir la orientación de las agujas del reloj*/
```

Rellenos

El relleno (o *padding* en inglés) nos permiten conocer el espacio entre la caja de un elemento y su contenido. Es parecido al margen pero ese espacio aplica desde el elemento hacia adentro, no hacia afuera. A continuación veremos varios ejemplos de uso y cómo su sintaxis es muy similar a los márgenes vistos previamente

Bordes

Hace referencia al borde un elemento, el cual se sitúa en medio del margen y el relleno. El borde requiere definir varias propiedades al mismo tiempo separadas con un espacio, como ser el grosor del borde (usualmente expresado en píxeles), el tipo de borde (como ser sólido o punteado) y el color del mismo. Como los márgenes y rellenos podemos definir el borde para los cuatro lados de la caja o para alguno en particular.

A continuación veremos varios ejemplos de uso:

```
h2 { border-top-color: blue; border-top-width: 2px; border-top-style: solid; }  
/*en este caso hemos puesto un borde superior de 2px de grosor, con un color azul y un estilo sólido*/  
  
div { width: 200px; height: 200px; border: 2px dotted red; }  
  
/*en este caso trabajamos con el shorthand , forma más sencilla de escribir el valor de la propiedad, decimos que los cuatro lados de este div tienen el mismo tipo de borde, de 2px dotted de estilo -un punto al lado del otro- y en color rojo*/  
  
#encabezado { width: 100%; height: 350px; border-left: 2px outset #f0f; }  
/*aquí expresamos que el borde de este elemento tendrá del lado izquierdo 2px de grosor, será un estilo outset -hacia afuera- y en color fucsia*/  
  
p { width: 300px; border: thin dashed rgba(0,255,0,0.5) border-top-color: red; }  
  
/*en este caso estamos diciendo que los cuatro lados de este párrafo al que le hemos aparte puesto un ancho de 300px será delgado -tenemos tres palabras thin, medium y thick aparte de medidas de longitud para expresar el grosor del borde- dashed de estilo -una linea al lado de la otra- y en color verde con 50% de alpha*/
```

Si quisiera eliminar el borde de algún elemento, lo único que debo hacer es escribir el borde al cual quiero afectar o border si quiero afectar a los cuatro lados y colocar el valor none.

Por ejemplo:

```
h1 { border: 1px solid red; border-top: none }
```

Sombra de caja

Esta propiedad se llama box-shadow y funciona de la siguiente forma.

```
a{ background-color: red; box-shadow: 2px 3px 5px #00f; }  
/* los vínculos tendrán una sombra que partirá hacia la derecha 2px hacia abajo 3px  
tendrá un difuminado de 5px -cuanto mayor valor más difuminado estará- y un color  
azul*/  
  
div { width: 300px; height: 200px; box-shadow: -2px -3px 10px 5px rgba(0,0,200,0.5);}  
/*este div tendrá una sombra hacia arriba de 3px hacia la izquierda de 2px -por esa  
razón se han colocado valores negativos, 10px de difuminado , 5px de distancia de  
sombra y con un color podríamos decir turquesa con 50% de alpha*/  
  
.varios { box-shadow: 1px -2px 4px inset;}  
/* los elementos cuya clase sea varios , tendrán una sombra hacia adentro en color  
negro - cuando no se especifica el color es negro- con un difuminado de 4px*/
```

Una forma fácil de entender esta propiedad es utilizar la herramienta online disponible en

<http://www.cssmatic.com/es/box-shadow>

Dónde podés crear tu propia sombra y luego sólo copiás y pegás el código en tus estilos por ejemplo

```
div {  
    box-shadow: 10px 10px 5px 0px rgba(0,0,0,0.75);  
}
```

Del sitio web, no copies los estilos que dicen -webkit- o -gecko- no son necesarios; sólo copia el último. Los otros son para navegadores viejos que ya no existen en el mercado

Propiedades del fondo de la caja

Color de fondo: background-color

Hace referencia al color de fondo de un elemento. Podemos aplicar esta propiedad a cualquier elemento que deseemos. Los valores posibles para esta propiedad son los mismos que la propiedad color (color de frente texto) , ya que los colores en la web siempre son iguales

Veamos algunos ejemplos de su uso:

```
div { width: 100%; height: 200px; background-color: blue; }
/*div con un color de fondo azul*/
header { width: 900px; background-color: red }
/*los header en general tendrán el color rojo de fondo*/

footer {width: 100%; height: 200px; background-color: green;}
/*los footer en general tendrán el color verde de fondo*/

#pie { height: 400px; background-color: rgba(255,0,0,0.5)}
/*el elemento cuyo id sea pie tendrá el color rojo de fondo con 50% de alpha*/
```

Imagen de Fondo: background-image

Esta propiedad me permite colocar una imagen de fondo en la caja. Siempre que trabajamos con un recurso que traemos a nuestra hoja de estilo (en este caso una imagen) necesitamos trabajar con una función llamada url, donde pondremos entre paréntesis la ruta relativa o absoluta de la imagen que queremos usar; por ejemplo:

```
div { width: 600px; height: 300px; background-image: url(imagenes/imagen.jpg)}
```

En este caso hemos trabajado con un formato de imagen JPEG, puede ser admitido también el formato PNG, GIF y SVG para trabajar.

Si queremos modificar el tamaño de la imagen de fondo, lo podemos hacer de la siguiente forma usando background-size:

```
div { width: 600px; height: 300px; background-image: url(imagenes/imagen.jpg);  
background-size: 10px 40px; }  
/*la imagen medirá 10px de ancho por 40px de alto*/  
div { width: 600px; height: 300px; background-image: url(imagenes/imagen.jpg);  
background-size: contain; }  
/*la imagen estará contenida, eso significa que mantendrá su proporción y sera  
colocada dentro del elemento sin dejar ninguna parte de esta por fuera, pero sí quizás  
dejando partes del elemento sin cubrir por la imagen*/  
  
div { width: 600px; height: 300px; background-image: url(imagenes/imagen.jpg);  
background-size: cover; }  
/*cover estira la imagen, incluso deformándola hasta cubrir el total del elemento  
que la tiene a esta de fondo*/  
  
div { width: 600px; height: 300px; background-image: url(imagenes/imagen.jpg);  
background-size: 30%; }  
/*la imagen en este ejemplo tiene 30% de ancho, al haber omitido la otra medida, eso  
significa que se mantendrá la proporción es decir no se deformará la imagen*/
```

También podemos decidir si queremos cambiar la forma en que se repite la imagen de fondo usando background-repeat, como vemos en los siguientes casos:

```
div { width: 600px; height: 300px; background-image: url(imagenes/imagen.jpg);  
background-repeat: no-repeat; }  
/*la imagen no se repetirá y aparecerá una sola vez en la esquina superior izquierda*/  
div { width: 600px; height: 300px; background-image: url(imagenes/imagen.jpg);  
background-repeat: repeat-x; }  
/*la imagen estará repetida hacia todo el ancho del elemento que la contiene*/  
  
div { width: 600px; height: 300px; background-image: url(imagenes/imagen.jpg);  
background-repeat: repeat-y; }  
/*la imagen estará repetida hacia todo el alto del que la contiene*/  
  
div { width: 600px; height: 300px; background-image: url(imagenes/imagen.jpg);  
background-repeat: repeat; }  
/*colocar este valor es innecesario pero lo hacemos para ilustrarlos mejor, es el valor  
predeterminado y significa que la imagen genera un tile o mosaico es decir se repite hacia  
el alto y el ancho del elemento que la contiene*/
```

Bordes Redondeados: border-radius

También podemos redondear las esquinas de un elemento usando border-radius y un valor en píxeles o porcentaje. Este valor especifica un radio de un círculo invisible que se dibuja sobre cada una de las esquinas y podemos definirlo para todas las esquinas al mismo tiempo o para algunas en particular.

```
a{ background-color: red; border-radius: 10px; }
/* las cuatro esquinas tienen un valor de radio de 10px*/

div { width: 300px; height: 200px; border-top-left-radius: 50%; }
/*en este div solo el borde superior izquierdo tiene 50% de radio, también podemos trabajar con border-top-right-radius, border-bottom-right-radius, y border-bottom-left-radius*/

.varios { border-radius: 3px 50% 20px; }
/* los elementos cuya clase sea varios, tendrán un radio superior izquierdo de 3px , un radio superior derecho de 50% , un radio inferior derecho de 20px y un radio derecho equivalente al radio superior derecho es decir 50%*/

#seccion { border-radius: 4px 5px; }
/*el elemento cuyo id sea seccion tendrá el borde superior izquierdo de 4px , el borde superior derecho de 5px y luego la equivalencia es diagonal es decir, el borde inferior izquierdo será igual al borde superior derecho y el borde inferior derecho igual al borde superior izquierdo*/
```

Si algo de esto te fue difícil de entender te voy a recomendar una aplicación online que te va a ayudar a generar tus propios bordes redondeados y demás propiedades de CSS

<http://border-radius.com/>

Lo único que te recomiendo es que deschequees las casillas que están debajo que dicen (Webkit o Gecko y solo dejá css3), te muestro un ejemplo

border radius

a service by jacob bijani

40

40

```
border-radius: 40px;  
border-bottom-right-radius: 100px;
```

WebKit Gecko CSS3

40

100

Si bien a veces termina generando quizás un poco más de código es una buena forma de aprender cómo crear los bordes redondeados en CSS.

Sólametno copialo y pegalo en tus estilos por ejemplo:

```
div {  
    border-radius: 40px;  
    border-bottom-right-radius: 100px;  
}
```

Organización de elementos con Flexbox

¿Cómo hacemos para organizar los elementos en columnas? porque hasta ahora hemos notado que nuestros contenedores siempre se colocan uno debajo del otro y a veces queremos lograr que los elementos compartan el espacio horizontal que hay disponible en un navegador.

Para eso vamos a usar una técnica de CSS conocida como Flexbox o cajas flexibles.

Comenzaremos por entender que todos los elementos, no importa cuántos éstos sean, deberán estar dentro de un elemento contenedor (como section, article , header, etc.) que tendrá aplicada la propiedad CSS display con el valor flex (si quiero trabajarla como elemento de bloque y tenga propiedades de caja) o inline-flex si quiero trabajarla como elemento de línea.

Ejemplo

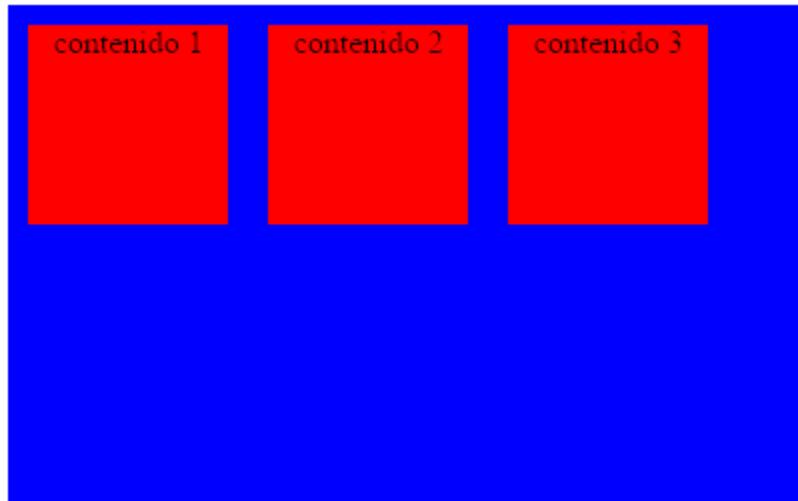
```
<div class="contenedor">
  <div class="contenidos">contenido 1</div>
  <div class="contenidos">contenido 2</div>
  <div class="contenidos">contenido 3</div>
</div>
```

En nuestra hoja de estilo deberíamos escribir lo siguiente:

```
.contenedor {
  display: flex;
  width: 400px;
  height: 250px;
  background-color:blue;
}

.contenidos {
  background-color: red;
  width: 100px;
  height: 100px;
  margin: 10px;
}
```

Esto se vería de la siguiente forma:



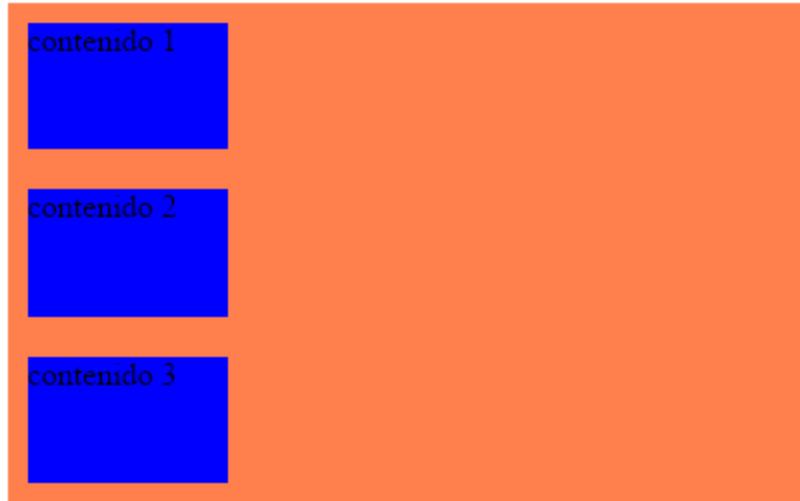
Hemos sumado la propiedad `text-align` con el valor `center`, dentro de la clase `contenidos` para alinear centralmente el texto como figura en la imagen.

Cambiando la dirección de los elementos

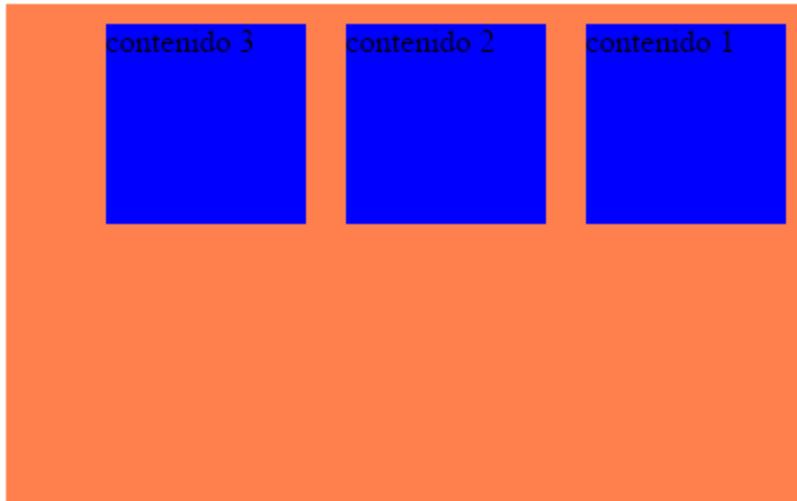
Por otro lado, Flexbox nos permite también cambiar la dirección de los elementos con la propiedad **flex-direction**. Ejemplo desde la hoja de estilo:

```
.contenedor {
    display: flex;
    flex-direction: row; /*valor predeterminado*/
    width: 400px;
    height: 250px;
    background-color:blue;
}

.contenidos {
    background-color: red;
    width: 100px;
    height: 100px;
    margin: 10px;
}
```



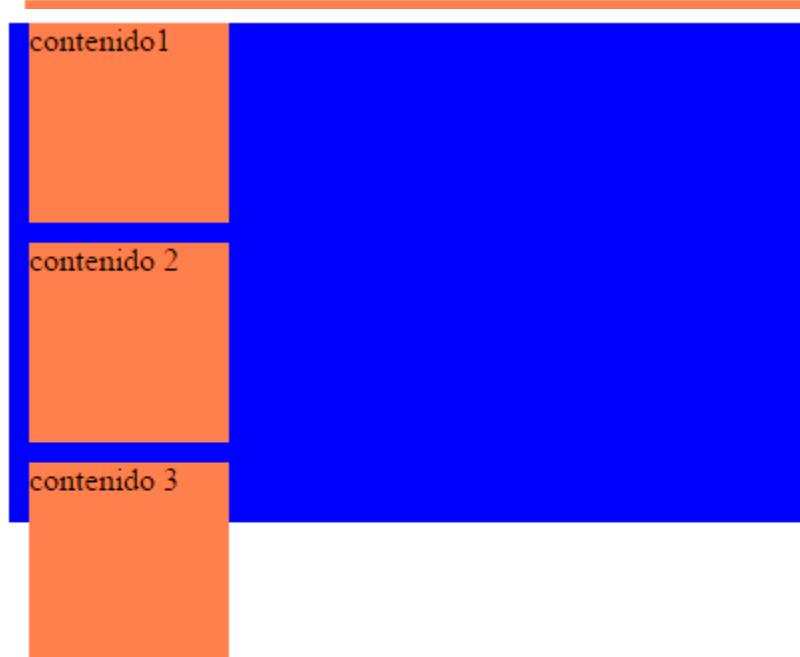
En nuestro navegador se verá como lo hemos visto hasta ahora, pero que pasaría si por caso el valor fuese en realidad **row-reverse**, se vería entonces de la siguiente manera:



Si el valor fuese **column**, se vería de la siguiente manera:

Y por otro lado **column-reverse** es exactamente igual, pero empezando por el último contenedor:

La verdad es que estos dos últimos valores parecieran que son como si no hubiésemos trabajado con Flexbox pero la realidad es que sí, nótese que diferente sería si no lo hubiésemos hecho:



En este último caso las cajas se desbordan, en cambio en el anterior no importa su alto siempre se adaptan a su contenedor.

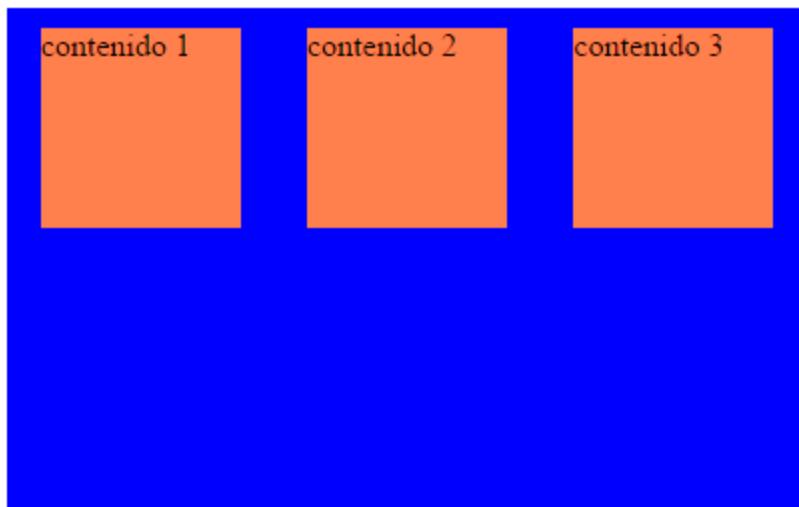
Espaciado entre elementos

Luego tenemos una propiedad que permite saber que se hace con el espacio entre los contenedores, es la propiedad **justify-content**, por ejemplo el valor predeterminado es **flex-start** eso significa como lo hemos visto que los elementos se sitúan al inicio del contenedor.

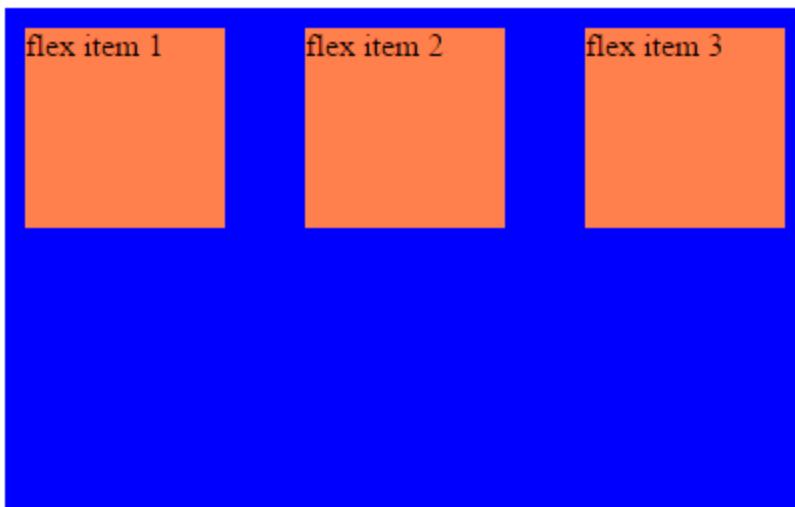
Por caso podemos trabajar también con el valor **space-between** (espacio entre ellos):

```
.contenedor {
    justify-content: space-between /* los elementos tienen espacio entre ellos*/
    display: flex;
    flex-direction: row; /*valor predeterminado*/
    width: 400px;
    height: 250px;
    background-color:blue;
}
```

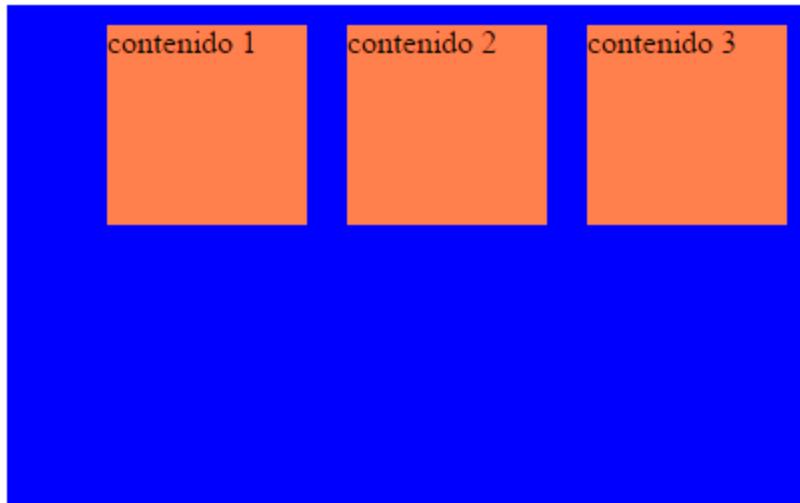
En el navegador se verá de la siguiente forma, notando que ahora los elementos se alinean con espacios entre ellos uniformemente para que ocupe todo el ancho del contenedor azul:



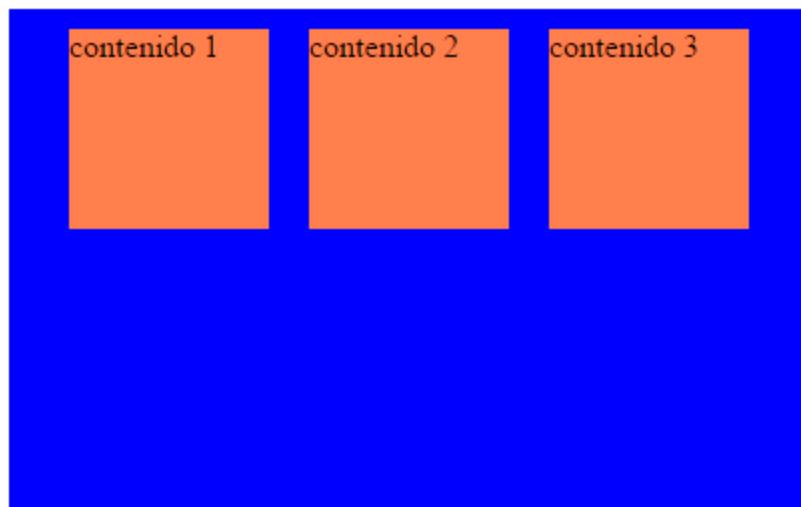
Luego también tenemos **space-around** que eso significa que habrá espacio entre los contenedores y al final y al principio de los mismos y se verá así:



El valor **flex-end** situará los elementos al final del contenedor general



Por último el valor **center**, los ubicará de la siguiente manera



Recordemos que además podemos manipular los elementos con márgenes, rellenos y otras propiedades que hemos aprendido anteriormente.

Generador online de Flexbox

Si te cuesta comprender todas las opciones de Flexbox te recomiendo una herramienta online que hará todo el trabajo sucio por vos: <http://css3generator.com/>

Simplemente seleccioná la opción Flexbox y mirá que simple panel te permitirá crear tu código CSS



Dándole estilos a los vínculos

Nos queda un elemento por remarcar que por demás es importante para poder dar un broche final a nuestro Sitio Web, el tema de los vínculos con CSS.

Veamos un ejemplo:

```
a {color: coral; }
```

Si a ese vínculo lo visito, no lo visito o si le paso el mouse por arriba siempre se verá así:

Usando pseudo-clases

vinculo

Podemos personalizar un poco nuestro vínculo usando pseudo-clases, que es una técnica en CSS para aplicar distintos estilos a varios estados de un elemento, como por ejemplo, un link ya visitado, o un elemento cuando paso con el mouse por encima sin tocarlo:

```
a:link {color: coral; }  
/*cuando no lo visité aún*/  
a:visited { color: green}  
/*cuando ya lo visité*/  
a:hover { color: red;}  
/*cuando le paso el mouse por arriba*/  
a:active { color: lightBlue}  
/*cuando le hago clic*/
```

Si por caso , queremos sólo afectar un solo vínculo o a varios y no a todos , lo haríamos de la siguiente forma definiendo un id único o una clase:

```
#vinculo:link {color: coral; }  
  
#vinculo:visited { color: green}  
  
.vinculos:hover { color: red;}  
  
.vinculos:active { color: lightBlue}
```

Quitar el subrayado

Desde que comenzó Internet, los links son automáticamente subrayados por el navegador para darle al usuario la indicación que se puede cliquear allí. Sin embargo, hay veces que podemos destacar el link de otra manera, ya sea con color u otro estilo y queremos eliminar ese subrayado. Veamos cómo se hace con **text-decoration: none;**

```
#vinculo:link {color: coral; text-decoration: none; }  
/* el valor predeterminado los vínculos, es de text-decoration: underline significa  
subrayado*/
```

FORMULARIOS

Un formulario es un contenedor con un conjunto de elementos que nos permite obtener información del usuario y mandarla a un servidor.

¿Servidor? ¿Qué es eso?

Entendiendo el modelo cliente-servidor

El servidor es lugar donde alojaremos nuestro Sitio Web más adelante. Es algo así como un alquiler que le pagamos a la empresa de *Hosting* o Alojamiento en inglés (la que nos dejará ocupar un espacio en su servidor) para entonces que luego cualquier persona que acceda desde Internet pueda ver nuestra Página Web. También hay lugares como Github que nos permiten alojar nuestro sitio web de forma gratuita.

El servidor es quién tiene la capacidad de darme respuestas o recibir peticiones (pedidos), y no solamente pueden estar en computadoras dedicadas a tales fines (como las que poseen las empresas de Hosting) sino que también puede crearse en nuestras propias computadoras siendo estos los conocidos servidores locales que comúnmente usamos para probar o testear elementos antes de subirlos a la web.

En el caso de formularios, el servidor recibe la información que recolectamos del usuario y la almacena localmente en una base de datos, una planilla de cálculo o la envía por correo electrónico a algún destinatario.

¿Qué es un formulario?

Más allá de esta aclaración lo importante, entender que es un formulario. Por caso, si yo entro a un buscador y pido toda la información relacionada con autos de carrera, la realidad es que no estoy ni más ni menos que un formulario de búsqueda, por ejemplo:

Buscar

Buscar

Si en todo caso le quiero por ejemplo quiero llenar un formulario de contacto que encontré en una página, estamos por ejemplo trabajando en un formulario de contacto por ejemplo:

Nombre:

Apellido:

Enviar

La realidad es que en el primero (formulario de búsqueda donde estamos buscando algo) obtenemos información a nuestro pedido y en el segundo brindamos información (nuestro nombre y apellido).

Un formulario suele ser un elemento HTML llamado <form> </form> con diversos otros elementos dentro de él que finalmente le dará la personalidad a este y determinará qué tipo de formulario será.

Componentes Básicos

Los formularios pueden contener muchos tipos de componentes interactivos dentro de él para que un usuario interactúe. Veamos los más importantes.

Campos de Ingreso de Texto

El campo de texto permite al usuario escribir cualquier tipo de texto sin ninguna restricción especial. Se define a través del elemento `<input>` con el atributo `type="text"`. Es el elemento que más vamos a ver en los formularios y seguramente del que nunca vayas a prescindir, por ejemplo:

```
<label>
  Nombre: <br>

  <input type="text">

</label>
```

Nótese que hay varios detalles en el ejemplo anterior, por un lado aparece el previamente trabajado campo de texto (el input) pero también otro elemento importante dentro de los formularios: el label, o etiqueta.

Etiquetas

El elemento **label** define un rótulo o etiqueta y nos permite identificar qué es lo que hay que escribir en el elemento de formulario del cual forma parte. Se ve de la siguiente manera en el navegador:

Nombre:

Podríamos llegar a preguntarnos: ¿por qué usar un label si en todo caso podría lograr lo mismo con un párrafo?

```
<p>
Nombre: <br>

<input type="text">

</p>
```

De hecho si esto mismo lo vemos en el navegador, se verá de la misma forma que usando un label. Entonces ¿cuál es la diferencia? El **label** tiene la funcionalidad de hacer foco en el elemento de formulario que le corresponda al momento de hacer clic con el mouse en él y de servir a los fines de accesibilidad. Por ejemplo, para una persona con dificultades visuales, una herramienta que lee la página en voz alta, sabe que en ese campo de texto hay que ingresar el Nombre.

Si hacemos clic en el `<label>` del primer ejemplo veremos como el cursor automáticamente empieza a titilar en el campo de texto como vemos en la siguiente imagen, aún si no cliqueamos dentro del campo de texto.

Y para que sirve esto?, justamente para lograr que el usuario pueda saber que ahí por caso debe completar su nombre y no en otro lugar.

Nombre:

Para que esto resulte, debemos trabajar con label de algunas de las siguientes formas:

```
<label>  
Nombre: <br>  
  
<input type="text">  
  
</label>
```

Anidando el componente dentro de la etiqueta

Este es el modo que usamos en el ejercicio anterior. En este caso el `<input>` se coloca dentro del `<label>` como si éste último sea un contenedor.

Utilizando el atributo for

Otra forma de relacionar un label con un elemento como input es a través del identificador de éste último sin necesidad de que uno esté dentro del otro, permitiéndonos ubicarlos separados en el diseño de la página. Para ello, el label posee un atributo **for** (para quién, en inglés) que nos permite definir para qué control estamos creando una etiqueta. Allí debemos usar el mismo identificador que hayamos usado en el `<input>`.

```
<label for="rotuloNombre">  
Nombre:  
</label>  
  
<input id="rotuloNombre" type="text">
```

En ambos casos logramos la misma funcionalidad que es que al hacerle clic a la palabra nombre en este caso, el cursor empieza a titilar (hace foco) en el campo de texto.

El Contenedor de Formulario

Este elemento anida a todos los elementos de nuestro formulario y posee una serie de atributos que son necesarios para que este luego funcione al momento de usar nuestro formulario, por ejemplo:

```
<form action="codigo.php" method="post">
<label> Nombre <br>
<input type="text">

</form>
```

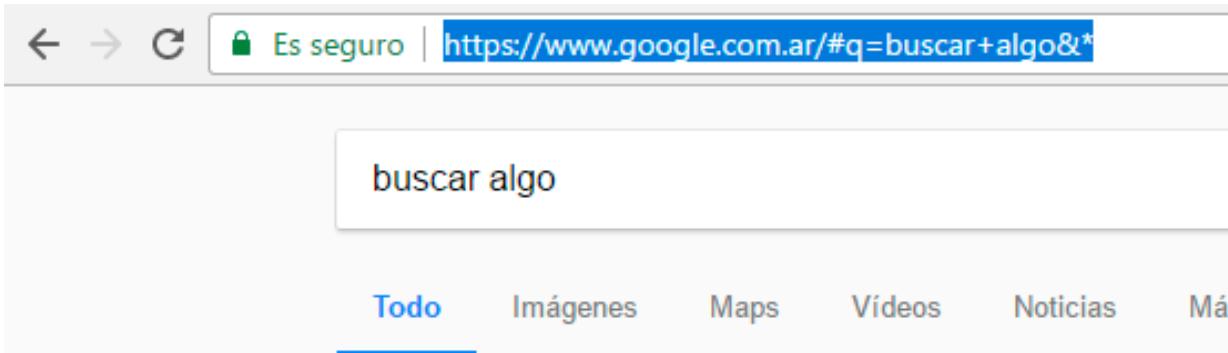
Acción del formulario

El atributo **action** (acción, en inglés) es la dirección de destino en el servidor que recibirá los datos del formulario. En general es una ruta (lugar donde está guardado algo) donde se encuentra un código de servidor que le dirá al formulario como actuar y qué hacer. Es importante que se entienda que como todo lo que vemos en el navegador lo estructuramos con HTML, pero que para que este formulario funcione (mande u obtenga información como hablamos antes) tiene que tener también relación con un lenguaje de servidor (un lenguaje que pueda ejecutarse en el servidor, cosa que HTML no puede hacer) como puede ser PHP , ASP, Node.js, y tantos más. Por esa razón en el ejemplo de la imagen hemos puesto un archivo llamado `codigo.php`, suponiendo que el servidor tiene un receptor programado en el lenguaje PHP llamado `codigo`.

Métodos del formulario

El método es el aquello que me dirá de qué forma se transmite la información. Al día de hoy se aceptan dos posibles metodologías para enviar la información al servidor y ellas son:

- **post**: Permite transmitir una gran cantidad de datos, es más seguro y permite adjuntar archivos.



- **get**: admite solo hasta 500 caracteres totales que haya tipeado el usuario y es un método que envía la información como apéndice de la URL, lo cual indica que lo que tipee el usuario en el formulario va a ser visto en la barra de direcciones del navegador.

Generalmente quien programe el servidor nos dirá cómo quiere que le enviemos los datos. En el siguiente ejemplo vemos que Google utiliza una técnica como get, dado que al buscar algo, veremos la búsqueda al final de la barra de direcciones.

En general podríamos decir que get es un método fácil de usar y muy útil en búsquedas y post se suele utilizar para el resto de los formularios.

Botones de Envío

Ningún formulario estará listo sin un botón para realizar el envío. Los botones se configuran con el mismo `<input>` que vimos antes pero con distintos tipos (**type**) según el tipo de botón. Los más comunes son el botón de texto y el de imagen.

Botón de Texto

```
<form action="codigo.php" method="post">
<label> Email <br>
<input type="email">
<input type="submit" value="Enviar">

</form>
```

Para crear un botón en texto usamos un input cuyo valor del type es **submit** (enviar en inglés), este botón también tiene un atributo llamado **value** que permite saber cuál será la leyenda en el botón,

es decir aquello que estará escrito en él. Recomendamos usar un verbo en infinitivo aquí para representar la acción como "Enviar", "Grabar", "Imprimir", "Comprar".

El botón de submit ejecuta el formulario, tomando todo lo que el usuario haya tipeado o ingresado en el formulario y lo envía al código.php (en nuestro ejemplo) que estaba como valor del action.

Otro elemento que también se dibuja como botón y envía nuestro formulario es el <button> </button>. Este elemento es muy utilizado porque nos permite colocar cualquier tipo de HTML dentro, o sea que puede ser texto, imagen, o cualquier combinación de HTML.

De todos modos en el navegador, también se verá igual que los otros botones.

```
<form action="codigo.php" method="post">
<label> Email <br>
<input type="email">
<button> Enviar </button>

</form>
```

Por último existe otro botón de texto que no tiene funcionalidades es decir no hace nada y sólo se ve como un botón. Este botón está pensado para ser programado con JavaScript y es el input con type="button". En este curso no lo usaremos dado que no hace nada *per se*.

```
<input type="button" value="Botón que no
hace nada">
```

Botón de Imagen

Si queremos que el botón sea una imagen que construimos gráficamente con íconos, fotos o botones animados, debemos usar el tipo **image** en el input.

```
<form action="codigo.php" method="post">
<label> Email <br>
<input type="email">
<input type="image" src="boton.jpg">

</form>
```

Como vemos en el ejemplo anterior, el botón de imagen hace lo mismo que el botón submit, pero nos permite incorporar una imagen a través del atributo src donde le decimos donde esta guardada ésta.

Componentes Avanzados

La mayoría de los elementos de nuestro formulario usarán la etiqueta **<input>** pero usaremos distintos valores para el atributo **type** (tipo en inglés). En el caso anterior usamos **text** (texto en inglés) veamos qué otros tipos hay.

Campo de Correo Electrónico

El campo de email, es un campo que nos permite validar un formato correcto de dirección de correo electrónico. Es decir que si yo escribo algo no válido o acorde a un email entonces no me dejará enviar el formulario y me alertará sobre tal cuestión.

```
<label>
Email: <br>

<input type="email">

</label>
```

Entonces, vamos a completar el ejemplo probemos hacer esto en el formulario escribiendo cualquier cosa por ejemplo un nombre o algo que no sea un correo electrónico válido:

Email

 ×

Debe escribir una dirección de correo electrónico válida

No en todos los navegadores se verá igual la validación; cada navegador tiene su estilo para validar el formulario. Esto no reemplaza a la validación que el programador del servidor va a tener que hacer de todas maneras para verificar que el dato esté correcto.

Campo de Dirección Web

El campo de URL valida el hecho de que el usuario ingrese un texto con un formato de dirección de web válida. Por ejemplo, si nos olvidamos de color http:// o nos olvidamos de terminar con .com o el dominio de alto nivel que fuere. También valida que no contenga espacios y caracteres que no están habilitados en una dirección Web.

Un detalle no menor es que este campo precisa de que uno anteponga el protocolo http:// o https:// para poder ser válido, por tanto es recomendable con el atributo **placeholder** también aplicable a este tipo de campo guiar al usuario para poder completarlo de manera adecuada. Por ejemplo:

```
<form action="codigo.php" method="post">
<label> Mi web Preferida
<input type="url" name="web" placeholder="http://www.dominio.com">
</label>
<button> Enviar </button>
</form>
```

Campo Numérico

El campo de número es un elemento que valida que el ingreso de texto sea un valor de campo que necesariamente debe ser un número. Por ejemplo, si yo escribo en el mismo una letra me dirá que es invalido el formato. En teléfonos celulares y tablets también mostrará un teclado numérico en pantalla en lugar del típico teclado con letras.

Este campo se define con un **type="number"** y soporta algunos atributos adicionales:

- **min**: Permite definir un valor mínimo que aceptamos. Si el usuario escribe un valor por debajo de este número será invalido el campo y el formulario no se enviará.
- **max**: Permite definir un valor máximo que aceptamos. Si el usuario escribe un valor por debajo de este número será invalido el campo y el formulario no se enviará.
- **step**: Permite definir un intervalo de valores posibles. Por ejemplo si queremos sólo números pares colocamos step="2", si aceptamos múltiples de 3 step="3"

Veamos entonces un ejemplo para un campo donde le pedimos al usuario ingresar su edad y suponemos edades de adultos válidas entre 18 y 100 años.

```
<form action="codigo.php" method="post">
<label>Edad
<input type="number" name="edad" min="18" max="100" >
</label>
<button> Enviar </button>
</form>
```

Campo de Teléfono

El campo de teléfono es útil en teléfonos móviles dado que nos pone un teclado numérico similar al de la aplicación de llamada donde podremos ingresar un número de teléfono. En computadoras no suele mostrar ninguna diferencia pero puede complementarse con un atributo llamado **pattern** que permite fijar un patrón que el usuario debe seguir sino el campo se considera invalido.

```
<form action="codigo.php" method="post">
<label>Telefono<br>
<input type="tel" name="telefono" pattern="
[+]\d{2}[\(]\d{2}[\)]\d{4}[-]\d{4}" placeholder="Formato:
+99(99)9999-9999" >
</label>
<button> Enviar </button>
</form>
```

El atributo **pattern** puede utilizarse en cualquier tipo de campo, y pueden colocarse diferentes tipos de patrones a seguir, conocidos como Expresiones Regulares, algo que está fuera del alcance de este curso pero que podés buscar en internet para ver más información.

Validación y Opciones Avanzadas

Ayuda de Campo de Formulario

Sobre un campo podemos agregar un texto de ayuda, conocido como **placeholder**.

```
<form action="codigo.php" method="post">
<label> Email <br>
<input type="email" placeholder="email@dominio.com" >
</label>
<button> Enviar </button>

</form>
```

El placeholder nos permite guiar al usuario, decirle qué completar y cómo dentro de nuestro campo, en este caso un campo de email tiene especificado el valor de información que debe

Email

email@dominio.com	<input type="button" value="Enviar"/>
-------------------	---------------------------------------

contener. Mientras el campo esté vacío veremos el texto de ayuda en color gris indicándonos cómo debemos ingresar el dato.

Campos Obligatorios

Cada campo en un formulario puede contener un atributo opcional **required** (obligatorio en inglés), cuya presencia sin valor nos permite hacer un campo obligatorio, es decir que si no se completa se mostrará un error como el siguiente en algunos navegadores cuando el usuario quiera enviar el formulario.

Email

Enviar

Éste es un campo necesario

El atributo obligatorio se agrega al campo como por ejemplo:

```
<input id="apellido" type="text" required>
```

Datos Múltiples

A través del atributo **multiple** podemos permitirle al usuario colocar más de un valor en campos de email y de archivos adjuntos, como por ejemplo varias direcciones de email seguida por coma, por ejemplo si en el navegador el usuario hace lo siguiente será perfectamente válido:

Email

Enviar

En el ejemplo anterior hemos agrandado el campo de email a través de la propiedad de CSS width, ejemplo:

Campos de Selección Nativos

HTML5 nos ofrece una selección de controles nativos, lo que significa que en lugar de dejarle al usuario tipar algo, le ofrecerá una interfaz gráfica para que con el mouse o el dedo en un equipo táctil elija o interactúe con un control más complejo.

Campos de Fecha y Hora

Los campos de fecha son un dato interesante pues hay de varios tipos y generan lo que comunmente vemos como un calendario.

```
<form action="codigo.php" method="post">
<label> Fecha
<input type="date" name="fecha" > </label>
<label> Hora
<input type="time" name="hora" ></label>
<label> Fecha y Hora
<input type="datetime" name="fecha y hora" > </label>
<label> Mes
<input type="month" > </label>
<label> Semana
<input type="week" > </label>
<button> Enviar </button>
</form>
```

En el ejemplo anterior se ve cómo podemos seleccionar el tipo de selector basado en el **type** del **<input>** solo fecha (**date**), hora (**time**), mes (**month**), semana (**week**) y también existe una variante de date y time (fecha y hora) llamada **datetime-local** que me permite fijar el calendario dando pauta de la zona horaria.

No todos los navegadores poseen todos los selectores. Por ejemplo, algunos teléfonos móviles no tienen selectores de mes o de semana y en esos casos simplemente mostrará un campo de texto normal para que el usuario ingrese un texto abierto.

Campos de Rango o Deslizadores

El campo de rango es ideal si queremos que el usuario indique cuánto le gustó una página, para elegir el volumen del audio o en líneas generales para calificar algo. Define un valor numérico entre un mínimo y un máximo pero donde el valor exacto no es tan importante, sino la proporción. Por ejemplo, un selector de volumen usualmente no nos dice si el valor está en 5.6 o 7.4, lo movemos con el dedo o mouse hasta que nos parece adecuado y no nos preocupamos por qué valor tiene.

Posee los mismos atributos que el campo de número, como **min**, **max** y **step**.

Veamos un selector de calificación donde el límite inferior será 0 y el superior 10.

```
<body>
<form action="codigo.php" method="post">
<label> Calificar la web del 0 al 10 <br>
<input type="range" min="0" max="10">
</label>
<button> Enviar </button>
</form>
</body>
```

En el navegador suele representarse con un slider o deslizador:

Calificar la web del 0 al 10



Campos de Color

El campo de color nos permite seleccionar un color en una paleta de colores que se despliega al hacerle clic al campo.

```
<body>
<form action="codigo.php" method="post">
<label> Color favorito <input type="color"
"> </label>
</label>
<button> Enviar </button>
</form>
</body>
```

El ejemplo anterior en el navegador se vería de la siguiente manera:



Campos de Búsqueda

El campo de búsqueda es un campo cuyo propósito es ser utilizado para realizar búsquedas, si bien parece un campo de texto, en algunos navegadores aparece al lado derecho una cruz para poder

borrar el contenido si el usuario quiere hacerlo, y guarda un historial de búsquedas pasadas en ese control.

```
<form action="codigo.php" method="post">
<label>Buscar <input type="search" autofocus>
</label>
</label>
<button> Enviar </button>
</form>
```

Es muy útil colocarle a este campo el atributo **autofocus**, este permite que automáticamente ni bien entramos ala página el cursor aparezca en este campo (haciendo foco automático).

Campos de Texto Multilínea

El campo de texto multilínea es un campo que permite escribir varias líneas a diferencia de un campo de texto `<input>`. Se utiliza para ello `<textarea>` y es obligatorio colocar la etiqueta de cierre `</textarea>`

```
<form action="codigo.php" method="post">
<label> Comentarios
<textarea name="comentario" maxlength="100"></textarea>
</label>
</form>
```

Nótese que hemos utilizado el atributo **maxlength** que permite fijar un máximo de caracteres a escribir.

Campos para Adjuntar Archivos o Fotos

El campo de archivo nos permite trabajar con la posibilidad de adjuntar uno o más archivos al formulario. Para usarlo es necesario que el formulario tenga el valor de **method** necesariamente en **post** y aparte se deberá agregar el atributo **enctype** como se ve en el siguiente ejemplo para que el servidor pueda recibir el archivo correctamente.

```
<body>
<form action="codigo.php" method="post" enctype="multipart/form-data">
<input type="file" multiple>
</form>
</body>
```

Si se ve en el ejemplo también se ha sumado el atributo **multiple** que previamente habíamos trabajado en el campo de email; si se coloca permite adjuntar múltiples archivos al mismo tiempo sino solamente se dejará trabajar con uno sólo.

En teléfonos móviles este campo se puede usar para que el usuario saque fotos con la cámara o selfies y las envíe el servidor.

Casillas de Verificación

Las casillas de verificación nos permite elegir una opción como verdadera o falsa, como por ejemplo si aceptamos un contrato o si queremos o no una cierta opción. Se utiliza también un <input> y el type **checkbox**.

```
<form action="codigo.php" method="post">
<label> Acepto Terminos y Condiciones
<input type="checkbox" name="acepto" checked>
</label>
</form>
</body>
```

Nótese que opcionalmente podemos usar el atributo **checked** (verificado, en inglés), este permite hacer que la casilla esté chequeada de manera predeterminada al cargar la página, el usuario puede deschequearla si así lo desea.

También en este campo es obligatorio el atributo **name** (nombre) que nos permitirá definir con un nombre interno qué es lo que el usuario está verificando.

Botones de Radio

Los botones de radio nos permiten elegir entre varias opciones sólo una de forma exclusiva. Por ejemplo Género, Estado Civil, cantidad de hijos, etc. Al marcar una opción, automáticamente se desmarcan las otras que sean parte del mismo tipo identificado por el mismo valor en el atributo **name** en el input de tipo **radio** como podemos ver en el siguiente ejemplo.

```
<form action="codigo.php" method="post">
<label> Casado
<input type="radio" value="casado" name="estadoCivil">
</label>
<label> Soltero
<input type="radio" value="soltero" name="estadoCivil">
</label>
</form>
```

Selectores de Lista

Los elementos de lista permite que el usuario elija opciones de una lista predefinida.

Sugerencias para Campos de Texto

El elemento **datalist** nos da la posibilidad de generar una lista de sugerencias que aparecerán como opciones de autocompletado en un campo de texto. En este control no obligatorio seleccionar un elemento de la lista, simplemente es una sugerencia; el usuario puede igual tipar otro valor.

Este elemento se relaciona con un campo de texto a través de dos atributos por un lado el atributo **list** en el **<input>** y por otro lado el atributo **id** del **<datalist>**.

```
<body>
<form action="codigo.php" method="post">
<input list="navegadores">
<datalist id="navegadores">
    <option value="Internet Explorer">
    <option value="Firefox">
    <option value="Chrome">
    <option value="Opera">
    <option value="Safari">
</datalist>
</form>
</body>
```

Dentro del datalist, creamos las opciones con el tag **<option>** que tiene el atributo value para el nombre de cada sugerencia que queremos ofrecer.

El ejemplo en el navegador se verá de la siguiente manera al cliquear en el campo de texto:



Selector de Lista

El elemento selector de lista me invita a elegir una opción entre varias, en este caso a diferencia del ejemplo anterior debo elegir sí o sí una de las opciones sugeridas; no es un campo abierto.

En este caso usamos un nuevo elemento: **<select>** con elementos **<option>** dentro.

```
<form action="codigo.php" method="post">
<select name="paises">
<option value="argentina"> Argentina </option>
<option value="brasil"> Brasil </option>
<option value="uruguay"> Uruguay </option>
</select>
</form>
```

Nótese que en el ejemplo anterior, se utiliza el atributo **value** para representar el valor de cada opción y el atributo **name** que ya fue utilizado en casos anteriores el cuál es necesario para transmitir el dato que selecciona el usuario al lenguaje de servidor que utilizará por ejemplo PHP. En el servidor entonces llegará que para el campo países se eligió por ejemplo Uruguay.

Elementos no interactivos

A veces tenemos que mostrar información que suele estar asociada a datos de un formulario o de una base de datos, pero no queremos que el usuario tenga la capacidad de modificar esta información. Para ello, HTML nos ofrece algunos elementos que si bien no son de formulario tienen relación con datos.

Medidor

El elemento **meter** (medidor en inglés) expresa gráficamente un valor conocido en un rango para saber cuán cerca del mínimo, máximo o valor deseado estamos, como un velocímetro en un auto.

Este elemento cuenta con varios atributos:

- value: El valor que queremos mostrar
- min: El valor mínimo aceptado
- max: El valor máximo aceptado
- high: El valor a partir del cual se considera que es alto
- low: El valor a partir del cual se considera que es bajo
- optimum: El mejor valor esperado

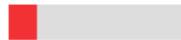
Veamos un ejemplo con resultados de una elección ficticia

```
<h1> Elecciones Resultados </h1>
<p> Candidato 1 </p>
<meter value="800" optimum="4000" max="5000" min="0" high="3500" low="1000">
</meter>
<p> Candidato 2 </p>
<meter value="3000" optimum="4000" max="5000" min="0" high="3500" low="1000">
</meter>
<p> Candidato 3 </p>
<meter value="4500" optimum="4000" max="5000" min="0" high="3500" low="1000">
</meter>
```

En el ejemplo anterior, se ve como hay tres candidatos y cuanto mayor el valor mejor va a ser, por tanto el optimum se fija entre el high y el max, este ejemplo en el navegador se verá de la siguiente manera.

Elecciones Resultados

Candidato 1



Candidato 2



Candidato 3



Barra de Progreso

La barra de progreso (etiqueta **progress**) nos permite indicar un progreso en una tarea o propósito.

```
<body>
<h1>Cantidad de KM recorridos </h1>
<progress value="200" max="1000"> </progress>
</body>
```

El **value** indica el valor actual de la tarea y el **max** fija el máximo que se desea cumplir, el ejemplo anterior en el navegador se verá de la siguiente manera.

Cantidad de KM recorridos



Agrupando secciones

En un formulario muchas veces tenemos varias zonas que agrupan controles, como puede ser: Datos Personales, Datos de Contacto, Datos Laborales, Domicilio y en cada grupo tenemos más de un campo. Para ello, HTML dispone de un contenedor llamado **fieldset** (conjunto de campos).

Cada conjunto de campos puede contener una leyenda o título que aparece visiblemente en el grupo usando la etiqueta **legend**.

Veamos un ejemplo:

```
<form method="post" action="codigo.php">
<fieldset>
<legend> Informacion Personal </legend>
<label> Nombre
<input type="text" name="nombre" required> </label>
<label> Email
<input type="email" name="email" multiple placeholder="mail@dominio.com">
</label></fieldset>
<fieldset>
<legend> Informacion Laboral </legend>
<label> A que se dedicaba en su empleo anterior
<textarea name="tareas" maxlength="2000"></textarea>
</label>
<p> Idiomas que habla </p>
<label> Ingles <input type="checkbox" name="idiomas" value="ingles"> </label>
<label> Francés <input type="checkbox" name="idiomas" value="frances"> </label>
<label> Español <input type="checkbox" name="idiomas" value="español"> </label>
</fieldset>
<button> Enviar </button>
</form>
```

En el caso anterior, hemos trabajado con **legend** en el **fieldset** para indicar una leyenda que el usuario verá y lo guiará en qué tipo de grupo de campos está trabajando.

Diseñando formularios

Con CSS podemos darle estilos a nuestros formularios como ya vimos anteriormente en el curso. También hay extensiones propias de formulario que nos permiten hacer cosas como darle algún tipo de característica particular al hecho de que un elemento es válido o inválido o si una casilla ha sido chequeada.

Pseudo-clases de validación

Para ello se usa un concepto de CSS conocido como pseudo-clases de validación que son selectores de CSS que nos permiten definir un estilo a aplicar sobre elementos en ciertas condiciones que dependen de las reglas de validación del formulario.

Por ejemplo:

```
<head>
<style>
input:invalid {
    border: 2px solid red;
}
input:valid {
    border: 2px solid green;
}
</style>
</head>
<body>
<h3>Selectores validos e invalidos</h3>
<form method="post" action="codigo.php">
<label> Mail <input type="email"></label>
<button> Enviar </button>
</form>
</body>
```

Si en el ejemplo anterior no completamos el campo o colocando un formato inválido de dirección de correo electrónico nos mostrará un borde en rojo; si por el contrario nuestro valor es correcto nos mostrará un color de borde verde.

También tenemos las pseudo-clases **:required** y **:optional** para diferenciar cuando un campo es obligatorio u opcional, **:focus** para identificar un campo que actualmente está en foco (con el cursor titilando dentro) y **:out-of-range** para identificar cuando un campo numérico tiene un valor fuera del rango permitido por las reglas de mínimo y máximo.

Para casillas de selección y botones de radio tenemos las pseudo-clases **:checked** y **:unchecked** para definir estilos para cuando está chequeada o no, respectivamente.

Para ayudarnos hay generadores online que nos facilitan la tarea como el que se encuentra en
<http://scriptgenerator.net/html-php-contact-form-mailer-generator/>

IMÁGENES Y VIDEOS

Las imágenes y los elementos gráficos son un elemento muy importante al momento de mostrarle algo al usuario que diga quiénes somos o qué estamos comunicando con nuestra página web.

Incorporando Imágenes

Repasso de Sintaxis img

Ya antes en este curso vimos cómo integrar una imagen a nuestro HTML:

```
<body>

</body>
```

También ya analizamos el funcionamiento de algunos atributos, como el src (la ruta del archivo de imagen que queremos colocar), alt (texto alternativo) y el ancho y alto.

Utilizando CSS podemos definir bordes y márgenes, como cualquier elemento de caja

```
<!DOCTYPE html>
<html>
<head>
<style>

img { width: 200px; height: 200px; border: 1px solid blue; margin-left: 5px; }

</style>
</head>
<body>

</body>
</html>
```

Para continuar de manera más profunda, vamos a ver que formatos es decir que tipo de elementos podemos incorporar como imágenes a nuestra página web.

Formatos de Gráficos

Ahora hablemos un poco más de los formatos de imágenes que podemos usar en una página web

Formato GIF

Al formato GIF se lo conoce como formato indexado eso significa que en realidad no tiene demasiada calidad fotográfica ya que utiliza un máximo de 256 colores que puede representar. Quizás era un buen formato cuando las conexiones no eran del todo buenas, o cuando los monitores no eran tan buenos para darnos cuenta de que claramente este formato no es el ideal , por ejemplo para transmitir áreas con degradados o fotografías de calidad.

El formato soporta el canal Alpha, lo que quiere decir "transparencia" y permite también trabajar con animaciones (los famosos GIF animados).

Formato JPEG

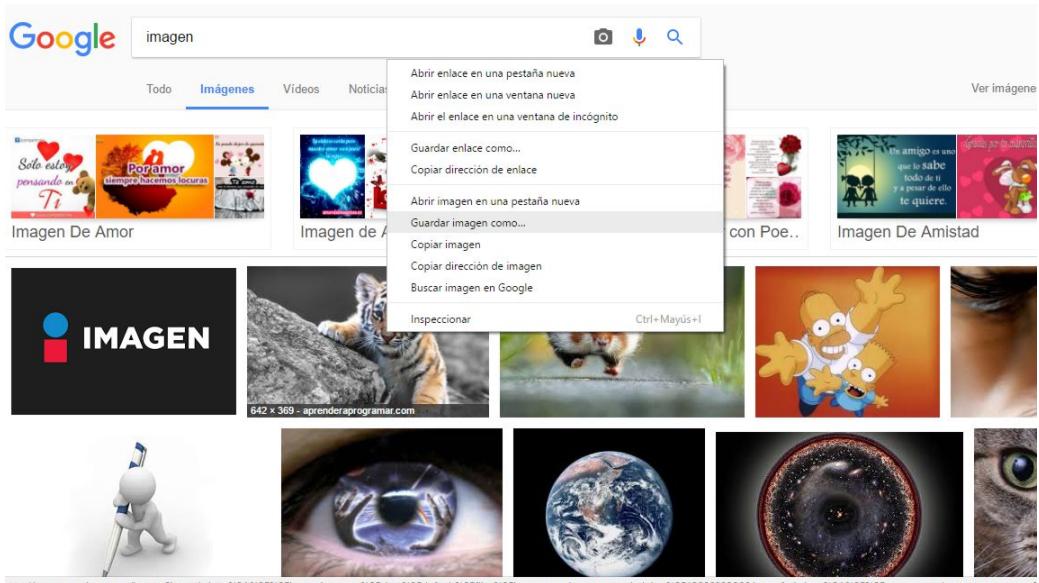
A diferencia del formato GIF, el JPEG (también conocido con su extensión de 3 letras JPG) contiene la posibilidad de almacenar fotos con millones de combinaciones de colores. La mayoría de las cámaras y teléfonos celulares, al sacar fotos, suelen trabajar con este formato. Es ideal para fotografías y no trabaja con el canal Alpha; es decir, es siempre rectangular y no se puede definir una zona transparente y no podremos nunca utilizarlo en un logo que queramos se funda con el fondo de nuestra página. Tampoco es útil para animaciones como si los son el formato GIF o el formato PNG.

Formato PNG

El formato PNG es un formato que puede reemplazar tanto al GIF como al JPG en diversos modos y calidades. Es hoy el formato más usado en la Web, tanto para logos, íconos y fotos dado que permite animaciones, transparencias, imágenes con pérdida de calidad y sin pérdida de calidad.

Creando Imágenes

Las imágenes que coloques en un sitio web pueden ser producto de las fotos que saques con tu cámara o teléfono celular, podés diseñarlas con un programa de edición de imágenes (como Adobe Photoshop o Adobe Illustrator), o podés buscar en repositorios públicos de imágenes, como puede ser Google Imágenes, accediendo a Google y seleccionando la pestaña "Imágenes". Desde allí podés guardar la imagen con click derecho, Guardar Imagen.



También otra opción que se activa al posar el mouse sobre una imagen que encontrás en la web y hacer clic en el botón derecho con la opción "Copiar URL de la imagen". Esto permite también trabajar con rutas absolutas en tu imagen, por ejemplo:

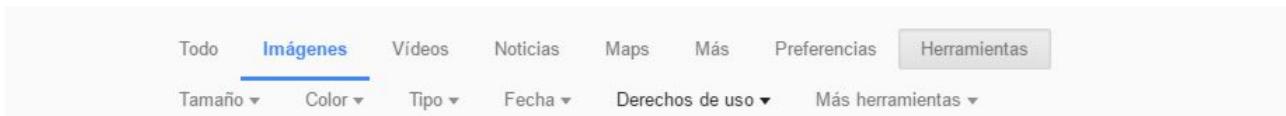


Vas a ver que sería algo así:

```

```

Es importante recordar que las imágenes podrían tener derechos de autor y no tener autorizada su uso en nuestro sitio web de forma gratuita. Siempre es conveniente revisar la licencia y términos del sitio de donde obtengamos las imágenes o buscar imágenes de libre distribución. Por ejemplo, en el buscador Google hay un filtro para poder detectar que imágenes con derechos autor o en qué podes usarlas:



Si querés editar imágenes o cambiar sus tamaños, no es necesario que conozcas herramientas profesionales o complejas te vamos a pasar algunas páginas para poder trabajar:

- <https://pixlr.com>
- <https://www.picmonkey.com/es>
- <https://lightroom.adobe.com>

El formato SVG

El formato SVG es un formato incorporado en HTML5. Los diseñadores y desarrolladores amamos este formato. ¿Por qué? Su sigla representa en inglés Scalable Vector Graphics (gráficos escalables vectoriales), y nos permite trabajar con un elemento que tiene todas las ventajas de un gráfico vectorial y también puede ser utilizado para la web.

Los gráficos vectoriales a diferencia de los otros gráficos como GIF o PNG, nunca pierden su calidad aun si hacemos zoom con los dedos en un teléfono móvil. En lugar de guardar información

de píxeles, un formato vectorial guarda información de cómo debe dibujarse el logo, ícono o gráfico, qué curvas lleva, qué color, etc. El hecho de ser vectorial también me permite trabajarla, editarla o mismo crearla en programas vectoriales como Adobe Illustrator o Corel Draw.

Usando el SVG embebido en HTML

SVG es un lenguaje basado en marcas como lo es HTML. Entonces qué tal si dibujamos algunas formas simples dentro de nuestro HTML con la etiqueta <svg>.

Para dibujar una forma circular debemos trabajar de la siguiente manera en el código:

```
<svg>
  <circle r="50" cx="20" cy="20" />
</svg>
```

El elemento <svg> actúa como contenedor en una página web de un gráfico SVG embebido en el HTML. Es imprescindible darle con estilos CSS, como por ejemplo:

```
<style>
  svg { width: 200px; height: 200px; border: 1px solid red; }
</style>
```

El otro detalle que se puede ver en el ejemplo es que estamos trabajando con varios atributos que no son de HTML sino de SVG:

- r: representa el radio del círculo en píxeles
- cx: es la ubicación del centro del círculo en el eje X u horizontal.
- cy: es la ubicación del centro del círculo en el eje Y o vertical

Para dibujar cuadrados en SVG, la historia es todavía más simple; usamos x, y para posición y width y height para las dimensiones ancho y alto..

```
<svg><rect width="200" height="200" x="10" y="20" > </svg>
```

Si por caso quisiéramos que este mismo rectángulo tenga las esquinas redondeadas tan solo deberíamos agregar los siguientes atributos, como veremos en el siguiente ejemplo de código

agregando rx y ry (radio del borde redondeado):

```
<svg><rect width="200" rx="20" ry="30" height="200" x="10" y="20" > </svg>
```

SVG con CSS

Esto es sólo una introducción al lenguaje SVG. Hay múltiples formas que podemos dibujar e incluso podemos darles estilos con CSS

En este ejemplo con stroke (trazo en inglés) hemos definido el color del borde de la elipse, con stroke-width hemos definido el grosor del trazo y con fill el color de fondo de nuestro elemento

```
<!DOCTYPE html>
<html>
<head>
<style>
svg { width: 200px; height: 200px; border: 1px solid red; }
.rectangulo { fill: blue; stroke: red; stroke-width: 12px; }
</style>
</head>
<body>

<svg><rect class="rectangulo" width="200" rx="20" ry="30" height="200" x="10" y="20" >
</svg>
```

Se ve que las propiedades utilizadas no son las típicas propiedades a encontrar en CSS para HTML sino que son propias de los elementos de SVG.

SVG a través de archivos externos

Otra de las formas de incorporar un SVGs a través de los siguientes elementos

```

<embed src="imagen.svg">
<iframe src="imagen.svg"> </iframe>
```

El elemento img ya conocido la incorpora de la misma manera que a cualquier otro formato de imagen. También vemos que se puede integrar como veremos en el siguiente ejemplo como imagen de fondo a través de la propiedad background-image.

```
<style>
div { background-image: url(imagenes/fondo.svg); }
</style>
```

También hay herramientas online para convertir imágenes en otros formatos a SVG, por ejemplo: <http://www.autotracer.org/es.html>

Pero si la idea es dibujar o generar de manera creativa elementos en SVG te recomiendo una página online que con simples opciones te permite generarlo: <http://www.svg-generator.de/>

Simplemente elegís los parámetros (el tamaño, cantidad, color). Un detalle importante es que está ahí el parámetro opacidad, o la transparencia, cuanto podés ver a través de ese elemento. El valor 100 es que el elemento no tienen transparencia el valor 0 es que tiene transparencia completa, si jugás en valores intermedios podés lograr cosas interesantes.



Luego cuando te guste lo que hiciste, hacés clic en "Export to SVG" se descarga a tu computadora y podés usarlo con las técnicas vistas anteriormente.

Trabajando con figuras y epígrafes

Cuando tenemos imágenes podemos darle una entidad mayor tratándolas como figuras (etiqueta `figure`). La figura es un contenedor que nos permite agrupar uno o más elementos gráficos bajo una entidad y poder luego asociarle un epígrafe (nota que explique la figura), por ejemplo:

```
<figure>
  
</figure>
```

Las figuras pueden incluir cualquier elemento gráfico explicativo por caso, en el siguiente ejemplo veremos cómo trabajar con `figure` y un elemento llamado `code` que sirve para mostrar código fuente dentro del navegador.

```
<body>
<figure>
  <pre>
    <code>
      selector { propiedad:valor; }
    </code>
  </pre>
</figure>
</body>
```

En el ejemplo, se utiliza esta etiqueta llamada `code` para poder enseñarles por ejemplo a alumnos como ustedes como se trabaja con una regla de estilo.

El elemento `figure`, también puede incluir un elemento llamado `figcaption`, que permite mostrar una explicación del elemento gráfico que contiene el elemento `figure` en cuestión, o epígrafe.

Veamos el siguiente ejemplo:

```
<body>
<figure role="group">
  <figcaption>Símbolos Patrios</figcaption>
  <figure>
    
    <figcaption>Escudo Nacional Argentino</figcaption>
  </figure>
  <figure>
    
    <figcaption>Bandera Nacional Argentina</figcaption>
  </figure>
</figure>
</body>
```

En el ejemplo anterior se ven varios puntos interesantes, se trabaja con **figure** que anida a su vez otros elementos figure lo cual es perfectamente válido. (recordemos que anidar significa colocar un elemento dentro de otro). Por otro lado, se trabaja con el **figcaption** en varias imágenes que están dentro de este figure principal, y al mismo **figure** principal se le asigna un atributo llamado role que determina de manera más completa el hecho de que este es el elemento **figure** padre que anida al resto.

Mapas de Imágenes

Los mapas de imágenes son elementos un poco más complejos que las imágenes que hemos visto anteriormente. Básicamente un elemento mapa de imagen está conformado de la siguiente manera:

```
<body>
  
  <map name="mapaMio">
    <area alt="Buenos Aires..." title="Buenos Aires"
          shape="rect" coords="683, 309, 754, 405"/>

    <area alt="Jujuy..." title="Jujuy"
          shape="rect" coords="200, 100, 0, 400"/>
  </map>
</body>
```

En este caso, el elemento **imagen**, tienen un atributo llamado **usemap** cuyo valor se relaciona con el identificador de un elemento llamado **map**.

Cada elemento **area** dentro del **map** define una zona cliqueable distinta dentro de la imagen. Por ejemplo, podemos tener un mapa de Argentina como una sola imagen y crear un mapa que defina un área por cada provincia que pueda tener un link a un HTML por provincia. En el ejemplo anterior hemos decidido generar dos rectángulos en las coordenadas en el primer caso 683 y 309 (esquina superior izquierda) y las coordenadas siguientes son de la esquina inferior derecha.

Esto permite que por caso en una misma imagen, tengamos varias áreas donde podemos hacer clic y por caso actuar como vínculos independientes más allá de ser partes de la misma imagen . Veamos el siguiente ejemplo:

```
<body>
  
  <map name="mapaMio">
    <area alt="Buenos Aires..." title="Buenos Aires"
          shape="rect" coords="683, 309, 754, 405" href="buenos-aires.html"/>

    <area alt="Jujuy..." title="Jujuy"
          shape="rect" coords="200, 100, 0, 400" href="jujuy.html"/>
  </map>
</body>
```

Dependiendo sobre qué área hagamos clic se nos llevará a la página jujuy.html o la página buenos-aires.html

En los ejemplos anteriores hemos trabajado con áreas en forma rectangular pero también podemos trabajar con otras formas o *shapes* en inglés por caso:

- **rect** (forma rectangular)
- **circle** (forma circular)
- **poly** (forma poligonal)

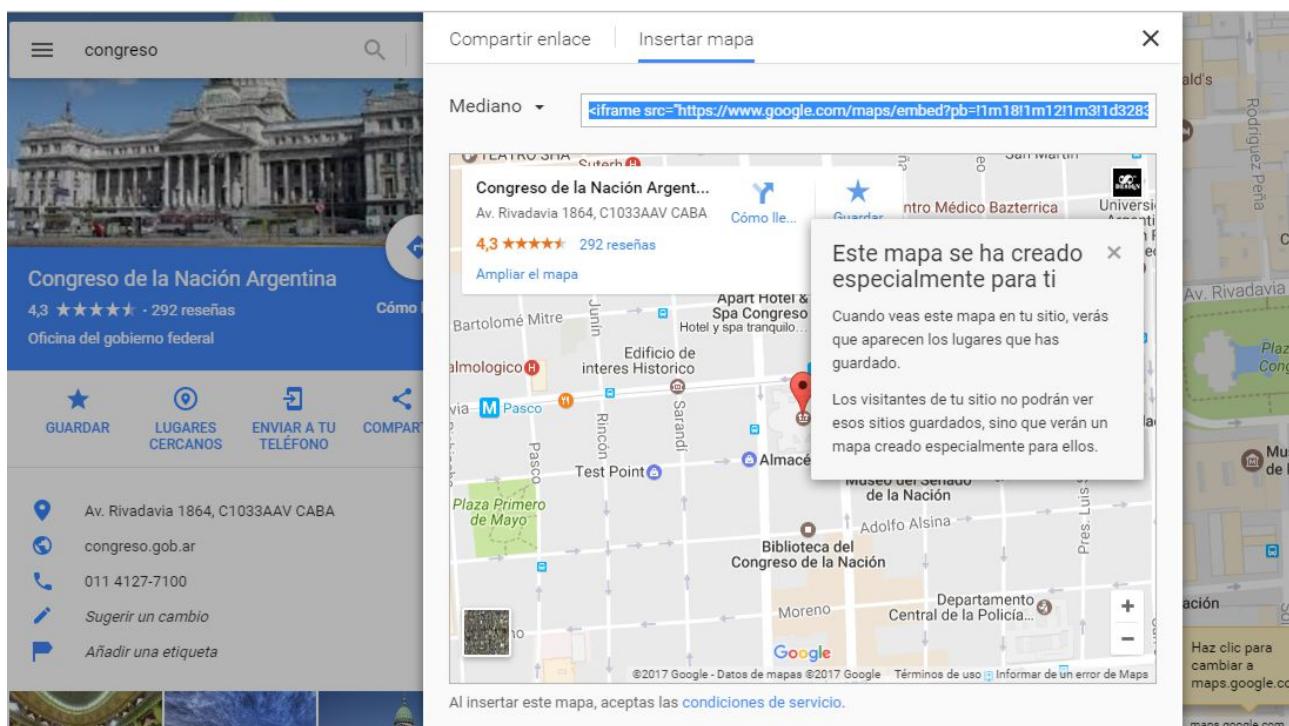
En cada caso se indican su coordenadas a través del atributo **coords** por ejemplo:

Hay programas que automáticamente nos dan la posibilidad de generar mapas y simplifican el trabajo, o también aplicaciones online que facilitan el trabajo, como por ejemplo www.image-maps.com.

Esta herramienta online es muy fácil de usar y podes generar el mapa que quieras. Lo que siempre hay que tener en cuenta es que esta página permite hacer elementos simples en segundos sin necesidad de conocer un programa de diseño demasiado avanzado.

Embeber otro contenido HTML con iframes

El iframe (cuadro interno) es como una ventana rectangular que ponemos en nuestra Web que muestra el contenido de otro HTML que puede estar alojado en otro servidor. Esto se usa para embeber en nuestro sitio contenido de fuentes externas, como puede ser un video de YouTube, una publicación de Facebook, un Tweet o un mapa de Google Maps. Veamos un ejemplo con Google Maps, obteniendo las coordenadas donde está el Congreso de la Nación en la Ciudad de Buenos Aires.



Cuando vamos a Google Maps podemos elegir la opción "Compartir" y luego la sección "Insertar Mapa". El código detallado en celeste en la imágenes el que ahora mostramos aquí:

```
<iframe src="https://www.google.com/maps/embed?pb=!1m18!1m12!1m3!1d3283.776355775465!2d-58.394794785252735!3d-34.60981636532688!2m3!1f0!2f0!3f0!3m2!1i1024!2i768!4f13.1!3m3!1m2!1s0x95bccac2cd859503%3A0x75cdafaaef2b946!2sCongreso+de+la+Naci%C3%B3n+Argentina!5e0!3m2!1ses!2sar!4v1491858376449" width="600" height="450" style="border: none" allowfullscreen></iframe>
```

Si vemos el código que Google Maps nos ofrece veremos que es una etiqueta iframe con algunos atributos, como:

- **src**: Indica la dirección del archivo se cargará en el iframe
- **allowfullscreen**: Me permite expandir o ver el elemento en pantalla completa si el usuario activa una opción desde dentro del contenido embebido
- **width**: Indica el ancho del iframe, puede indicarse también en porcentaje
- **height**: Indica el alto del iframe, puede indicarse también en porcentaje

Los iframes incluyen un marco con un borde por defecto, podemos quitarlo con CSS usando un estilo en línea con la propiedad **border: none;**.

Podés probar crear un iframe desde un video de YouTube o desde un mensaje de Facebook usando la opción de Embeber y copiando y pegando el código con iframe que estos sitios nos ofrecen.

Incluyendo contenido multimedia

HTML5 nos permite además de colocar imágenes en una página web, incorporar un reproductor de audio y un reproductor de video.

Reproduciendo Audio

El elemento audio desde nuestro código se verá de la siguiente manera

```
<audio src="audio.mp3" controls loop autoplay preload="metadata"></audio>
```

La etiqueta audio posee los siguientes atributos, los cuales no son todos obligatorios:

- **src**: Indica la ruta donde está guardado el archivo de audio que quiero escuchar, usualmente un archivo MP3
- **loop**: Si está presente define una reproducción indefinida del audio salvo que el usuario pare la misma con el botón de pausa
- **autoplay**: Si está presente nos da la posibilidad de que le audio se reproduzca ni bien carga la página sin necesidad de hacerle clic al botón de "Play" (esta opción en algunos navegadores no funciona por considerarlo intrusivo)
- **controls**: Si está presente muestra los controles de reproducción (Play, Pausa, control de volumen y la barra de progreso de la reproducción). En cada navegador el diseño de estos controles puede variar.
- **preload**: Si está presente, permite acelerar la reproducción del archivo descargando los primeros bytes. De esta forma, al presionar Play iniciaría inmediatamente. Los posibles valores son: a) **metadata**, donde sólo descarga los datos de tipo de archivo, nombre de la canción, autor, y calidad; b) **auto**: descarga los meta datos y además algunos segundos del contenido; c) **none**: no hace ninguna precarga.

Reproduciendo Video

En el caso del video el sistema es similar al de audio, lo único que cambia es que añadimos dos atributos más y se usa la etiqueta <video>:

```
<video src="video.mp4" poster="imagen.jpg" width="300" controls muted loop autoplay> tu navegador no soporta este elemento </video>
```

A los atributos de audio, agregamos:

- **poster**: define una imagen que actúa como poster en lugar de un recuadro negro o gris hasta que la reproducción del video comience. Usualmente se coloca una imagen representativa del video.
- **width** y opcionalmente **height**: define el ancho y alto del reproductor de video
- **muted**: permite silenciar el video. Si está definido controls, el usuario puede subir el volumen manualmente.



Personalizar botones de reproducción

Si queremos personalizar los botones de reproducción debemos recurrir necesariamente a JavaScript que está fuera del ámbito de este curso, pero te dejo un ejemplo para que veas

```
<video id="video" poster="imagen.jpg" > tu navegador no
soporta este elemento
<source src="video.ogg">
<source src="video.mp4">

</video>
<button onclick="reproducir()"> Play </button>

<script>
function reproducir(){
document.getElementById('video').play()
}
</script>
```

En el navegador este ejemplo se verá de la siguiente forma:



De todas formas puede que lo anterior no sea sencillo y siempre tenemos herramientas para poder realizar nuestros elementos de HTML, en este caso en esta sencilla herramienta podemos trabajar simplemente generando de manera automática nuestra etiqueta de video: <http://scriptgenerator.net/simple-html5-embed-video-player/>

Dibujando con Canvas

El canvas, o lienzo en inglés, es un elemento gráfico muy interesante pues permite generar juegos o interfaces ricas y animadas a través de JavaScript.

Si bien es un poco complejo para quien no maneja JavaScript o algún lenguaje de programación, vamos a mostrar algunos puntos básicos e importantes sobre este elemento de HTML.

```
<!DOCTYPE html>
<html>
<head>
    <title>Canvas</title>
<style>
canvas { width: 200px; height: 200px; border: 1px solid red; }

</style>
</head>
<body>
<canvas> </canvas>

</body>
</html>
```

El elemento canvas define una zona rectangular (cuyas dimensiones definimos con CSS) con un color de fondo (blanco por defecto) que no tiene ningún dibujo dentro.

A través de JavaScript podremos entonces dibujar utilizando primitivas de dibujo, como el siguiente ejemplo:

```
<body>
  <canvas id="miCanvas" width="578" height="200"></canvas>
  <script>
    var canvas = document.getElementById('miCanvas');
    var context = canvas.getContext('2d');
    //LUEGO IRÁ EL CÓDIGO SEGÚN EL DIBUJO A REALIZARSE
  </script>
</body>
```

Ahora por ejemplo, dibujemos un cuadrado dentro del canvas.

```
<body>
  <canvas id="myCanvas" width="578" height="200"></canvas>
  <script>
    var canvas = document.getElementById('myCanvas');
    var context = canvas.getContext('2d');

    context.beginPath();
    context.rect(188, 50, 200, 100);
    context.fillStyle = 'yellow';
    context.fill();
    context.lineWidth = 7;
    context.strokeStyle = 'black';
    context.stroke();
  </script>
</body>
```

En el ejemplo anterior se ha dibujado un rectángulo en la coordenada 188 en cuanto al eje x y 50 en cuanto al eje y del canvas, luego tiene un ancho de 200px por un alto de 100px, con un relleno o color de fondo en amarillo y finalmente un borde de 7px de ancho de color negro.

Ahora veamos cómo generar un círculo

En el ejemplo anterior, el círculo tiene un radio de 70px y también observamos donde se ubica a través de las variables centerX y centerY que centran el círculo en la mitad del canvas. Las variables en JavaScript son algo así como cajones donde guardamos información que luego reutilizaremos.

Se indica que el círculo tendrá un color de relleno en verde y un color de borde #003300.

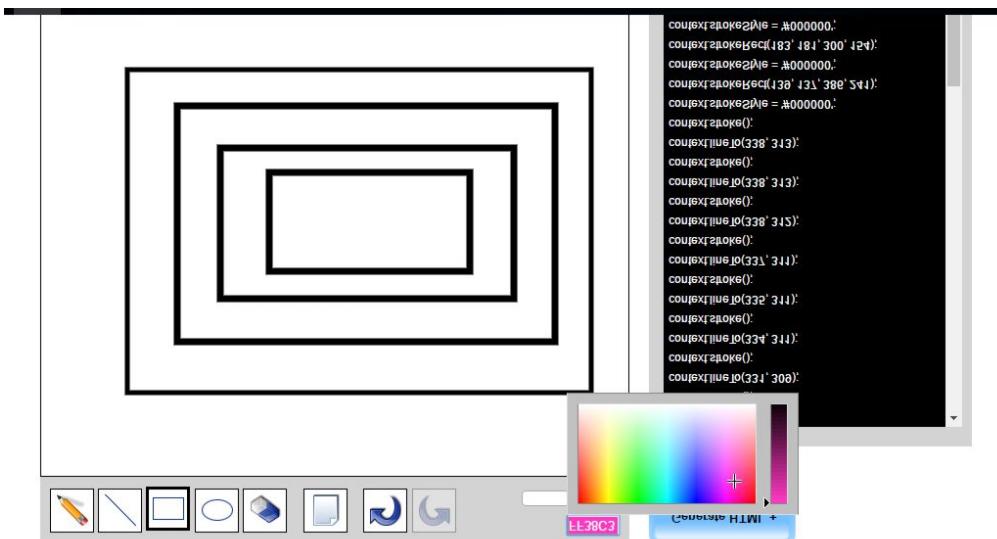
```

<script>
    var canvas = document.getElementById('myCanvas');
    var context = canvas.getContext('2d');
    var centerX = canvas.width / 2;
    var centerY = canvas.height / 2;
    var radius = 70;

    context.beginPath();
    context.arc(centerX, centerY, radius, 0, 2 * Math.PI, false);
    context.fillStyle = 'green';
    context.fill();
    context.lineWidth = 5;
    context.strokeStyle = '#003300';
    context.stroke();
</script>

```

Como muchas otras características de HTML, existen editores online como <http://www.htmlcanvasstudio.com/>. A través de un simple panel puedes determinar la forma, los colores y copiar el código en tu HTML



CONCLUYENDO

Con este tema finalizamos este curso de HTML y CSS. Como te darás una idea, el tema de diseño web es enorme y hay mucho más por aprender. Lo importante no es buscar la perfección desde el principio y animarte a publicar en Internet algo que hayas desarrollado.

Podés publicar un diseño web que estés realizando en sitios como GitHub, Google Firebase Static Hosting, InfinityFree o ProFree Host.

También podés contratar un servicio de alojamiento que es como un alquiler y un dominio (como .com o .com.ar) para así tener un servicio más profesional, que hoy en día se consiguen de forma muy económica. Una vez que tengas tu sitio web online y funcionando, podrás aprender mucho más de diseño y de programación web.

¡Bienvenido o bienvenida al apasionante mundo del diseño web, previo paso a la programación web!