**HTML5**

Cuando se están escribiendo documentos HTML5, una de las primeras nuevas características nuevas que usted notará es la declaración del DOC TYPE

<!DOCTYPE HTML>

La codificación de caracteres (charset) también ha sido simplificada

<meta charset="UTF-8">

**Nuevos elementos en HTML5**

<article>, <aside>, <audio>, <canvas>, <datalist>, <details>, <embed>, <footer>, <header>, <nav>, <output>, <progress>, <section>, <video> …

La codificación de caracteres por defecto en HTML5 es UTF-8.

**Novedades en HTML5**

Formularios

- La especificación de Formularios Web 2.0 permite la creación de formularios más potentes y con experiencias de usuario más enriquecidas.

- Selectores de fecha, selectores de color y controles de paso numéricos han sido agregados.

- Los tipos de campo de entrada ahora incluyen email, búsqueda y URL.

- Los métodos de formulario PUT y DELETE son ahora soportados.

API Integrada (Interfaces de Programación de Aplicaciones)

- Arrastrar y soltar

- Audio y Video

- Aplicaciones Web sin conexión

- Historial

- Almacenamiento Local

- Geolocalización

- Mensajería Web

**Lista de Modelos de Contenido**

En HTML, los elementos pertenecían típicamente al modelo de contenido de bloque o contenido en línea. HTML5 introduce siete modelos de contenido principales.

- Metadata

- Embedded (Incrustado)

- Interactive (Interactivo)

- Heading (Encabezado)

- Phrasing (Expresión)

- Flow (Flujo)

- Sectioning (Seccionamiento)

Los modelos de contenido HTML5 son diseñados para hacer la estructura de marcado más útil para el explorador y el diseñador web.

**Modelo de contenido**

Metadata: Contenido que define la presentación o comportamiento del resto del contenido. Estos elementos son encontrados en el head (la cabecera) del documento.

Elementos: <base>, <link>, <meta>, <noscript>, <script>, <style>, <title>

Incrustado: Contenido que importa otros recursos al documento.

Elementos: <audio>, <video>, <canvas>, <iframe>, <img>, <math>, <object>, <svg>

Interactivo: Contenido específicamente destinado para interacción del usuario.

Elementos: <a>, <audio>, <video>, <button>, <details>, <embed>, <iframe>, <img>, <input>, <label>, <object>, <select>, <textarea>

Encabezado: Define un encabezado de sección.

Elementos: <h1>, <h2>, <h3>, <h4>, <h5>, <h6>, <hgroup>

Expresión: Este modelo tiene un número de elementos en línea en común con HTML4.

Elementos: <img>, <span>, <strong>, <label>, <br />, <small>, <sub>, y más.

El mismo elemento puede pertenecer a más de un modelo de contenido.

Flujo de contenido: Contiene la mayoría de los elementos HTML5 que serían incluidos en el flujo normal del documento.

Seccionamiento de contenido: Define el alcance de encabezados, contenido, navegación y pies de página.

Elementos: <article>, <aside>, <nav>, <section>

Los distintos modelos de contenido se sobreponen en ciertas áreas, dependiendo de cómo están siendo usados.

**Estructura de pagina HTML5.**

Una estructura de página genérica HTML5 se ve así:



**El elemento <header>**

En HTML5, una simple etiqueta <header> es usada.

El elemento <header> es apropiado para ser usado dentro de la etiqueta body.

<!DOCTYPE html>

<html>

<head></head>

<body>

<header>

<h1>Encabezado más importante</h1>

<h3>Encabezado menos importante</h3>

</header>

</body>

</html>

Note que la etiqueta <header> es completamente diferente de la etiqueta <head>.

**El elemento <footer>**

El elemento footer también es utilizado ampliamente. Generalmente, al hablar del footer nos referimos a una sección localizada en la parte inferior de la página web.

<footer>…</footer>

La siguiente información se provee usualmente dentro de estas etiquetas:

- Información de Contacto

- Política de Privacidad

- Íconos de redes sociales

- Términos de Servicio

- Información de Derechos de autor

- Mapa del sitio y documentos relacionados

**El elemento <nav>**

Esta etiqueta representa una sección de una página que conecta a otras páginas o ciertas secciones dentro de la página. Esto sería una sección con hipervínculos de navegación

<nav>

<ul>

<li><a href="#">Home</a></li>

<li><a href="#">Servicios</a></li>

<li><a href="#">Sobre nosotros</a></li>

</ul>

</nav>

No todos los hipervínculos en un documento deberían estar dentro de un elemento <nav>. El elemento <nav> está hecho sólo para bloques grandes de hipervínculos de navegación. Típicamente, el elemento <footer> tiene una lista de hipervínculos que no necesitan estar en un elemento <nav>.

**El elemento <article>**

El elemento article es una pieza autónoma, de contenido independiente que puede ser usada y distribuida separadamente del resto de la página o sitio. Esto podría ser un post de un foro, una revista o un artículo de revista, una entrada de blog, un comentario, un widget interactivo o gadget, o cualquier otra pieza de contenido independiente.

<article>

<h1>El título del Artículo</h1>

<p>Contenidos del elemento artículo</p>

</article>

Cuando un elemento <article> es anidado, el elemento interior representa un artículo relacionado al elemento exterior. Por ejemplo, comentarios de un post en un blog pueden ser elementos <article> anidados en el <article> representando el post del blog.

**El elemento <section>**

<section> es un contenedor lógico de la página web o artículo. Las secciones pueden ser usadas para dividir contenido dentro de un artículo.

Por ejemplo, una página principal podría tener una sección para introducir a la compañía, otra para ítems de noticias, y también otra para la información de contacto.

Cada <section> debería ser identificada, típicamente incluyendo un encabezado (elemento h1-h6) como un hijo del elemento <section>.

<article>

<h1>Bienvenido</h1>

<section>

<h1>Encabezado</h1>

<p>contenido o imagen</p>

</section>

</article>

Si tiene sentido ubicar separadamente el contenido de un elemento <section>, use mejor un elemento <article>.

**El elemento <aside>**

El elemento <aside> es contenido secundario o tangencial que podría ser considerado separado pero indirectamente relacionado al contenido principal.

Este tipo de contenido es a veces representado en barras laterales. Cuando una etiqueta <aside> es usada dentro de una etiqueta <article>, el contenido de <aside> debería estar específicamente relacionado a ese artículo.

<article>

<h1> Regalos para todos</h1>

<p>Este sitio web será el mejor lugar para escoger regalos </p>

<aside>

<p>Los regalos serán entregados a usted dentro de las siguientes 24 horas </p>

</aside>

</article>

Cuando una etiqueta <aside> es usada fuera de una etiqueta <article> su contenido debería estar relacionado al contenido que está a su alrededor.

**Audio en la web**

Antes de HTML5, no había un estándar para reproducir archivos de audio en una página web. El elemento HTML5 <audio> especifica un estándar para instructar audio en una página web.

Hay dos maneras diferentes de especificar la url de origen del audio. El primero usa el atributo source:

<audio src="audio.mp3" controls>

Elemento de audio no soportado por su explorador

</audio>

La segunda forma usa el elemento <source> dentro del elemento <audio>

<audio controls>

<source src="audio.mp3" type="audio/mpeg">

<source src="audio.ogg" type="audio/ogg">

</audio>

Múltiples elementos <source> pueden ser enlazados a diferentes archivos de audio. El explorador usará el primer formato reconocido.

El elemento <audio> crea un reproductor de audio dentro del explorador.

<audio controls>

<source src="audio.mp3" type="audio/mpeg">

<source src="audio.ogg" type="audio/ogg">

Elemento de Audio no soportado por su explorador.

</audio>

El texto en medio de las etiquetas <audio> y </audio> se mostrará en los exploradores que no soporten el elemento <audio>.

**Atributos de <audio>**

controls

Especifica qué controles de audio deberían ser desplegados (como el botón de play/pause etc.)

autoplay

Cuando este atributo es definido, el audio empieza a reproducirse tan pronto está listo, sin preguntar por el permiso del visitante.

<audio controls autoplay>

loop

Este atributo es usado para que el audio se vuelva a reproducir cada vez que finaliza.

<audio controls autoplay loop>

Actualmente, hay tres formatos de archivo soportados para el elemento <audio>: MP3, WAV y OGG.

**Videos en html**

El elemento video es similar al elemento audio. Se puede especificar la URL de origen del video usando un atributo en un elemento de video, o usando elementos source dentro del elemento video:

<video controls>

<source src="video.mp4" type="video/mp4">

<source src="video.ogg" type="video/ogg">

Elemento de video no es soportado en su explorador

</video>

Otro aspecto que los elementos de audio y video tienen en común es que los exploradores más importantes no soportan los mismos tipos de archivo. Si el explorador no soporta el primer tipo de video, intentará el siguiente.

**Atributos de <video>**

Otro aspecto compartido por los elementos audio y video es que cada uno tiene atributos controls, autoplay y loop.

En este ejemplo, el video volverá a empezar después de que termine de reproducirse:

<video controls autoplay loop>

<source src="video.mp4" type="video/mp4">

<source src="video.ogg" type="video/ogg">

Elemento de Video no es soportado en su explorador

</video>

Actualmente, hay tres formatos de video soportados para el elemento <video>: MP4, WebM, y OGG.

**Barra de progreso**

El elemento <progress>(progreso) provee la habilidad de crear barras de progreso en la web.

Este elemento progreso puede ser usado con encabezados, párrafos, o en cualquier otra parte del body.

Atributos del Elemento de Progreso

Value: Especifica cuánto de la tarea ha sido completado.

Max: Especifica cuánto trabajo la tarea requiere en total.

Status: <progress min="0" max="100" value="35"></progress>

Usa la etiqueta <progress> en conjunto con JavaScript para mostrar dinámicamente el progreso de una tarea.

**Almacenamiento WEB HTML**

Con almacenamiento web HTML5, los sitios web pueden guardar datos en el computador local de los usuarios. Antes de HTML5, teníamos que usar Cookies de JavaScript para alcanzar esta funcionalidad.

Las ventajas del Almacenamiento Web

- Más seguro

- Más rápido

- Guarda una cantidad más grande de datos.

- Los datos guardados no son enviados con cada petición del servidor

El almacenamiento local es por dominio. Todas las páginas desde un dominio pueden guardar y acceder a los mismos datos.

**Tipos de objetos de Almacenamiento Web**

Hay dos tipos de objetos de almacenamiento web:

- sessionStorage()

- localStorage()

Local vs. Session

- El almacenamiento por sesión es destruido una vez el usuario cierra el explorador.

- El almacenamiento local guarda datos sin fecha de expiración.

Necesita estar familiarizado con JavaScript básico para entender y usar la API.

**Usando Valores**

La sintaxis para el almacenamiento web en modo local y en modo sesión es muy sencilla y similar.

Los datos son guardados como parejas de clave/valor.

Guardando un valor:

localStorage.setItem("clave1", "valor1");

Obteniendo un valor:

//esto imprimirá el valor

alert(localStorage.getItem("clave1"));

Removiendo un valor:

localStorage.removeItem("clave1");

Removiendo todos los valores:

localStorage.clear();

La misma del almacenamiento por sesión es casi igual, con una diferencia: En vez de localStorage se usa sessionStorage.

**¿Qué es la API de GeoLocalización?**

En HTML5, la API de GeoLocalización es usada para obtener la localización geográfica del usuario.

Ya que esto puede comprometer la privacidad del usuario, la opción no está disponible a menos que el usuario la apruebe.

La geolocalización es mucho más precisa para dispositivos con GPS, como teléfonos inteligentes y demás.

**Usando Geolocalización HTML**

El método principal de la API de Geolocalización es getCurrentPosition, que obtiene la localización geográfica actual del dispositivo del usuario.

navigator.geolocation.getCurrentPosition();

Parámetros:

showLocation (obligatorio): Define el método callback que obtiene la información de localización.

ErrorHandler(opcional): Define el método callback que es invocado cuando un error ocurre procesando la llamada asincrónica.

Options (opcional): Define un conjunto de opciones para obtiene la información de localización.

Es necesario estar familiarizado con JavaScript básico para entender y usar la API.

**Presentando los Datos**

La localización del usuario puede ser presentada de dos formas: Geodetic y Civic.

1. La forma geodetic para describir posición se refiere directamente a latitud y longitud.
2. La representación civic de los datos de localización es presentada en un formato que es leído más fácilmente y entendido por la persona promedio.

Cada parámetro tiene una representación tanto geodetic como civic.



El método getCurrentPosition() devuelve un objeto si su respuesta es exitosa. Las propiedades de latitud, longitud y exactitud son siempre retornadas.

**Haciendo los objetos arrastrables**

La característica de arrastrar y soltar le permite "agarrar" un objeto y arrastrarlo a un lugar diferente.

Para hacer un elemento arrastrable, sólo establece el atributo draggable a verdadero:

<img draggable="true" />

Cualquier elemento HTML puede ser arrastrable.

La API para arrastrar y soltar en HTML está basada en eventos.

<!DOCTYPE HTML>

<html>

<head>

<script>

function allowDrop(ev) {

ev.preventDefault();

}

function drag(ev) {

ev.dataTransfer.setData("text", ev.target.id);

}

function drop(ev) {

ev.preventDefault();

var data = ev.dataTransfer.getData("text");

ev.target.appendChild(document.getElementById(data));

}

</script>

</head>

<body>

<div id="box" ondrop="drop(event)"

ondragover="allowDrop(event)"

style="border:1px solid black;

width:200px; height:200px"></div>

<img id="image" src="sample.jpg" draggable="true"

ondragstart="drag(event)" width="150" height="50" alt="" />

</body>

</html>

Qué Arrastrar

Cuando el elemento es arrastrado, el atributo ondragstart llama a una función, drag(event), que especifica qué información será arrastrada.

El método dataTransfer.setData() establece el tipo de datos y el valor de los datos arrastrados.

function drag(ev) {

ev.dataTransfer.setData("text", ev.target.id);

}

En nuestro ejemplo, el tipo de datos es "texto" y el valor es el ID del elemento arrastrable ("image").

Dónde Soltar

El evento ondragover especifica dónde los datos arrastrados pueden ser soltados. Por defecto, los datos y elementos no pueden ser soltados en otros elementos. Para permitir soltar datos, debemos prevenir el manejo por defecto del evento.

Esto se hace llamando al método event.preventDefault() para el evento event.

Al Soltar

Cuando los datos arrastrados se sueltan, ocurre un evento drop.

function drop(ev) {

ev.preventDefault();

var data = ev.dataTransfer.getData("text");

ev.target.appendChild(document.getElementById(data));

}

El método preventDefault() previene al explorador de manejar los datos con su opción por defecto (por defecto se abre como un evento link on drop)

Los datos arrastrados pueden ser accedidos con el método dataTransfer.getData(). Este método retornará cualquier dato que fue configurado del mismo tipo en el método setData(). Los datos arrastrados son el ID del elemento arrastrado ("image").

Al final, el elemento arrastrado es anexado en el elemento soltado, usando la función appendChild().

Se requiere un conocimiento básico de JavaScript para entender y usar la API.

**Dibujando Formas**

SVG significa Gráficos Vectoriales Escalables (Scalable Vector Graphics), y es usado para dibujar formas con estilo de marcado HTML.

SVG ofrece varios métodos para dibujar trazos, rectángulos, círculos, texto e imágenes.

SVG no está basado en píxeles, por lo que puede ser magnificado infinitamente sin pérdida de calidad.

**Insertando Imágenes SVG**

Una imagen SVG puede ser agregada al código HTML con solo una etiqueta básica de imagen que incluye un atributo source apuntando a la imagen:

<img src="image.svg" alt="" height="300" />

SVG define imágenes basadas en vectores en formato XML.

**Dibujando un círculo**

Para dibujar formas con SVG, primero necesitamos crear una etiqueta de elemento SVG con dos atributos: ancho y alto.

<svg width="1000" height="1000"></svg>

Para crear un círculo, agregamos una etiqueta <circle>:

<svg width="2000" height="2000">

<circle cx="80" cy="80" r="50" fill="green" />

</svg>

- cx mueve el centro del círculo más a la derecha de la pantalla

- cy mueve el centro del círculo más abajo respecto a la parte superior de la pantalla

- r define el radio

- fill determina el color de nuestro círculo

- stroke agrega un marco al círculo

Todos los elementos y atributos de los archivos SVG pueden ser animados.

**Otros Elementos de formas**

<rect> define un rectángulo:

<svg width="2000" height="2000">

<rect width="300" height="100"

x="20" y="20" fill="green" />

</svg>

Los atributos width y height del elemento <rect> definen la altura y la anchura del rectángulo.

<line> define un segmento de línea:

<svg width="400" height="410">

<line x1="10" y1="10" x2="200" y2="100"

style="stroke:#000000; stroke-linecap:round;

stroke-width:20" />

</svg>

(x1, y1) define las coordenadas de inicio(x2, y2) define las coordenadas finales.

<polyline> define formas construídas de múltiples definiciones de líneas:

<svg width="2000" height="500">

<polyline style="stroke-linejoin:miter; stroke:black;

stroke-width:12; fill: none;"

points="100 100, 150 150, 200 100" />

</svg>

**<ellipse> y <polygon>**

**Elipse**

El elemento <ellipse> es similar al elemento <circle>, con una excepción:

Puedes modificar independientemente los ejes horizontales y verticales de su radio, usando los atributos rx y ry.

<svg height="500" width="1000">

<ellipse cx="200" cy="100" rx="150" ry="70" style="fill:green" />

</svg>

**Polígono**

El elemento <polygon> es usado para crear un gráfico con al menos tres lados. El elemento polígono es único porque cierra la forma automáticamente por usted.

<svg width="2000" height="2000">

<polygon points="100 100, 200 200, 300 0"

style="fill: green; stroke:black;" />

</svg>

Polygon viene del griego. "Poly" significa "muchos" y "gon" significa "ángulo".

**Animaciones de Forma**

Las animaciones SVG pueden ser creadas usando el elemento <animate>.

El siguiente ejemplo crea un rectángulo que cambiará su posición en 3 segundos y luego repetirá la animación dos veces:

<svg width="1000" height="250">

<rect width="150" height="150" fill="orange">

<animate attributeName="x" from="0" to="300"

dur="3s" fill="freeze" repeatCount="2"/>

</rect>

</svg>

attributeName: Especifica cuál atributo será afectado por la animación

from: Especifica el valor de comienzo del atributo

to: Especifica el valor final del atributo

dur: Especifica cuánto tiempo corre la animación (duración)

fill: Especifica si el valor del atributo debe o no devolverse a su valor inicial cando la animación finaliza (Valores: "remove" resetea el valor; "freeze" conserva el “to value”)

repeatCount: Especifica el conteo de repetición de la animación

En el ejemplo anterior, el rectángulo cambia su atributo x de 0 a 300 en 3 segundos.

Para repetir la animación indefinidamente, utiliza el valor "undefined" para el atributo repeatCount.

**Trazos**

El elemento <path> es usado para definir un Trazo.

Los siguientes comandos están disponibles para datos de Trazo:

M: mover hacia

L: línea hacia

H: línea horizontal hacia

V: línea vertical hacia

C: curva hacia

S: curva suave hacia

Q: curva Bézier cuadrática

T: curva Bézier cuadrática suave

A: Arco elíptico

Z: closepath, cerrar trazado.

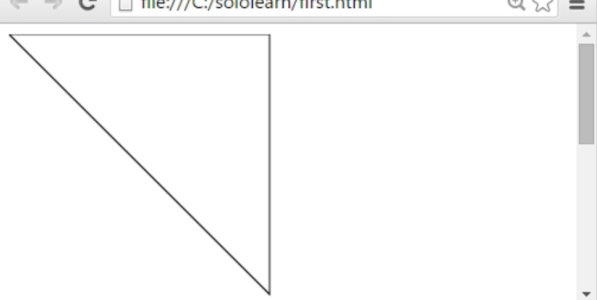
Defina un trazo usando el atributo d:

<svg width="500" height="500">

<path d="M 0 0 L200 200 L200 0 Z" style="stroke:#000; fill:none;" />

</svg>

M coloca nuestro "lápiz virtual" abajo en la posición 0,0. Luego se mueve 200px hacia abajo y a la derecha, luego se mueve hacia arriba a la posición 200,0. El comando Z cierra la forma, lo que resulta en una hipotenusa:



Todos los comandos presentados arriba pueden también ser expresados con letras minúsculas. Cuando se usan letras mayúsculas, se indica una posición absoluta; minúsculas indican una posición relativa.

**El elemento <canvas>**

El canvas HTML es usado para dibujar gráficos que incluyen desde líneas simples a objetos gráficos complejos.

El elemento <Canvas> es definido por:

<canvas id="canvas1" width="200" height="100"></canvas>

El elemento <canvas> es solo un contenedor para gráficos. Se debe usar un script para dibujar realmente los gráficos (usualmente JavaScript).

El elemento <canvas> debe tener un atributo id para que pueda ser referido por JavaScript:

<html>

<head>

<script>

var can = document.getElementById("canvas1");

var ctx = can.getContext("2d");

</script>

</head>

<body>

<canvas id="canvas1"

width="400" height="300"></canvas>

</body>

</html>

getContext() retorna un contexto de dibujo sobre el Canvas.

Conocimiento básico de JavaScript es requerido para entender y usar el Canvas.

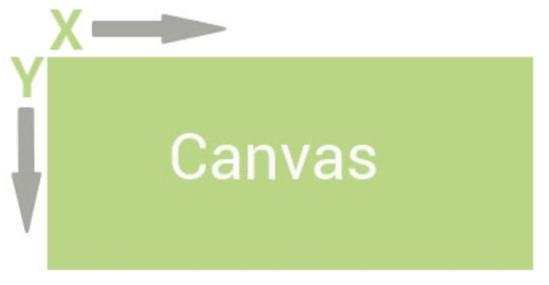
**Coordenadas del Canvas**

El Canvas HTML es una grilla bi-dimensional.

La esquina superior izquierda del Canvas tiene las coordenadas (0,0).

La coordenada X aumenta hacia la derecha.

La coordenada Y aumenta hacia la parte inferior del Canvas.



El elemento <canvas> es solo un contenedor para gráficos.

**Dibujando Formas**

El método fillRect(x, y, w, h) dibuja un rectángulo "relleno", en el cual w indica el ancho y h indica altura. El color de relleno por defecto es negro.

Un rectángulo de 100\*100 pixeles es dibujado en el Canvas en la posición (20, 20):

var c=document.getElementById("canvas1");

var ctx=c.getContext("2d");

ctx.fillRect(20,20,100,100);

La propiedad fillStyle es usada para establecer un color, gradiente, o patrón para llenar el dibujo.

Usar esta propiedad le permite dibujar un rectángulo con relleno verde.

var Canvas=document.getElementById("canvas1");

var ctx=canvas.getContext("2d");

ctx.fillStyle ="rgba(0, 200, 0, 1)";

ctx.fillRect (36, 10, 22, 22);

El Canvas soporta varios métodos varios para dibujar:

Dibujar una línea

moveTo(x,y): Define el punto de inicio de la línea.

lineTo(x,y): Define el punto final de la línea.

Dibujar un círculo

beginPath(): Inicia el dibujo.

arc(x,y,r,start,stop): Establece el parámetro del círculo.

stroke(): Finaliza el dibujo.

Gradientes

createLinearGradient(x,y,x1,y1): Crea un gradiente lineal.

createRadialGradient(x,y,r,x1,y1,r1): Crea un gradiente radial/circular.

Dibujando texto sobre el Canvas

Font: Define las propiedades de la fuente para el texto.

fillText(text,x,y): Dibuja texto "relleno" sobre el Canvas.

strokeText(text,x,y): Dibuja texto sobre el Canvas (sin relleno).

Hay muchos otros más enfocados a ayudar a dibujar formas e imágenes sobre el Canvas.

**Canvas vs. SVG**

**Cavas:**

- Los elementos se dibujan programáticamente

- Los dibujos se realizan con píxeles

- Las animaciones no están incorporadas

- Alto rendimiento para operaciones de dibujo basadas en píxeles

- Dependiente de la Resolución

- No hay soporte para controladores de eventos

- Puede guardar la imagen resultante como .png o .jpg

- Muy adecuado para los juegos de gráficos intensivos

**SVG:**

- Los elementos son parte del DOM de la página (Modelo de Objetos del Documento)

- El dibujo se hace con vectores

- Efectos, como animaciones son incorporadas

- Basado en sintaxis XML estándar, que provee mejor accesibilidad.

- Independiente de la Resolución

- Soporte para manejadores de eventos

- No apto para aplicaciones de juegos

- Es mejor para aplicaciones con áreas grandes de renderización (por ejemplo, Google Maps)

Podemos realmente usar tanto SVG como Canvas en la misma página, de ser necesario.

De todas formas, usted no puede dibujar SVG en un Canvas, o viceversa.

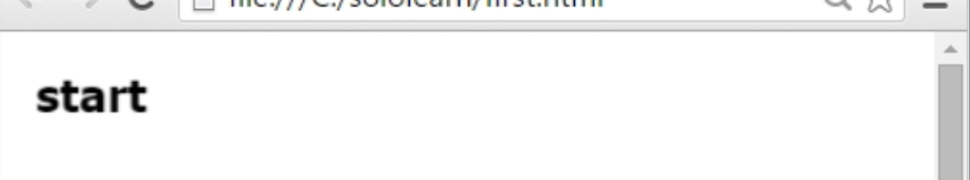
**Trabajando con Canvas**

El elemento Canvas puede ser transformado. Por ejemplo, un texto es escrito en el Canvas en las coordenadas (20, 10).

ctx.font="bold 22px Tahoma";

ctx.textAlign="start";

ctx.fillText("start", 10, 30);



El método translate(x,y) es usado para mover el Canvas.

X indica qué tan lejos mover la grilla horizontalmente, y Y indica qué tan lejos mover la grilla verticalmente.

ctx.translate(100, 150);

ctx.fillText("after translate", 10, 30);

En este ejemplo, el Canvas se mueve 100px hacia la derecha, y 150px hacia abajo.



Canvas tiene varios métodos para dibujar rutas, cajas, círculos, texto y añadir imágenes.

**El método rotate()**

El método rotate() es usado para rotar el Canvas en HTML5. El valor debe ser en radianes, no en grados.

Acá hay un ejemplo que dibuja el mismo rectángulo antes y después de fijar la rotación:

ctx.fillStyle = "#FF0000";

ctx.fillRect(10,10, 100, 100);

ctx.rotate( (Math.PI / 180) \* 25); //rotate 25 degrees.

ctx.fillStyle = "#0000FF";

ctx.fillRect(10,10, 100, 100);



La rotación sólo afectará a los dibujos hechos después de que la rotación se haya realizado.

**El método scale()**

El método scale() escala el dibujo actual. Recibe dos parámetros:

- El número de veces en que la imagen debería ser escalada en la dirección X.

- El número de veces en que la imagen debería ser escalada en la dirección Y.

var canvas = document.getElementById('canvas1');

ctx =canvas.getContext('2d');

ctx.font="bold 22px Tahoma";

ctx.textAlign="start";

ctx.fillText("inicio", 10, 30);

ctx.translate(100, 150);

ctx.fillText("después de trasladar", 0, 0);

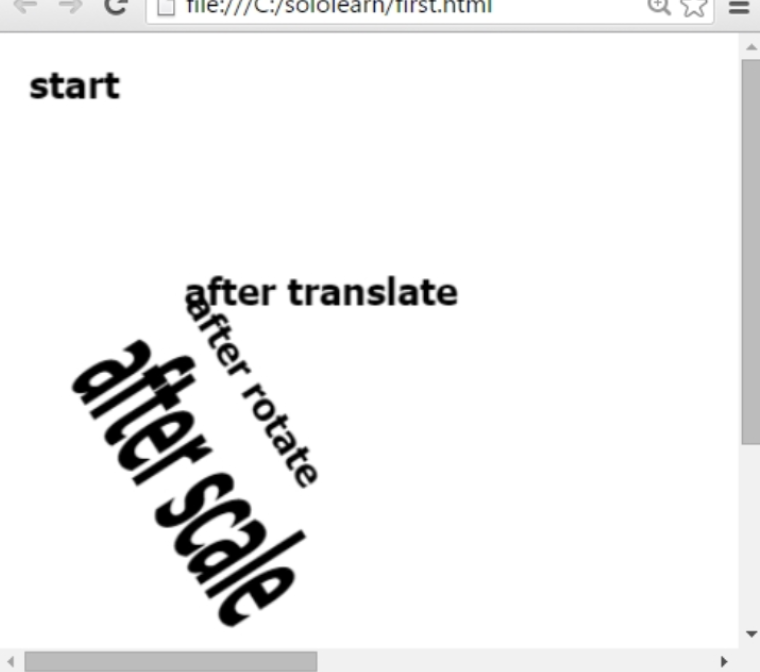
ctx.rotate(1);

ctx.fillText("después de rotar", 0, 0);

ctx.scale(1.5, 4);

ctx.fillText("después de escalar", 0,20);

Esto escalará el canvas 1.5 veces en la dirección X, y 4 veces en la dirección Y:



Si cambias la escala de un dibujo, todos los futuros dibujos también sufrirán este cambio.

**Formularios HTML5**

HTML5 trae muchas características y mejoras a la creación de formularios web. Hay nuevos atributos y tipos de entradas que fueron introducidos para ayudar a crear mejores experiencias para los usuarios web.

<form>

<label>Su nombre:</label>

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

</form>

Utiliza el atributo novalidate para evitar la validación de formularios en envíos.

**Nuevos Atributos**

HTML5 ha introducido un nuevo atributo llamado placeholder. En los elementos <input> y <textarea> este atributo provee una indicación al usuario de qué información puede ser ingresada en el campo.

<form>

<label for="email">Your e-mail address: </label>

<input type="text" name="email"

placeholder="email@example.com" />

</form>

El atributo autofocus hace que el cursor se posicione en el campo de entrada deseado cuando el formulario se carga:

<form>

<label for="e-mail">Your e-mail address: </label>

<input type="text" name="email" autofocus />

</form>

El atributo required dice al navegador que el campo de entrada es obligatorio.

**Formularios con campos requeridos**

El atributo "required" es usado para crear los elementos de entrada necesarios:

<form autocomplete="off">

<label for="e-mail">Su e-mail: </label>

<input name="Email" type="text" required />

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

</form>

El formulario no será enviado sin llenar los campos requeridos.

El atributo autocomplete especifica si un formulario o campo de entrada debería tener autocompletar activado o no.

Cuando autocompletar está activado, el explorador automáticamente completa valores basado en valores que el usuario ha ingresado antes.

HTML5 agregó varios tipos de entrada nuevos:

- color

- date

- datetime

- datetime-local

- email

- month

- number

- range

- search

- tel

- time

- url

- week

Nuevos atributos de entrada en HTML5:

- autofocus

- form

- formaction

- formenctype

- formmethod

- formnovalidate

- formtarget

- height and width

- list

- min and max

- multiple

- pattern (regexp)

- placeholder

- required

- step

Los tipos de entrada que no son soportados por exploradores web desactualizados, se comportarán como una entrada de texto.

**Creando un cuadro de búsqueda**

El Nuevo tipo de entrada search puede ser usado para crear un cuadro de búsqueda:

<input id="mysearch" name="searchitem" type="search" />

Acuérdate de asignar un nombre a tu entrada, de lo contrario no se enviará nada.

**Opciones de búsqueda**

La etiqueta <datalist> puede ser usada para definir una lista de opciones pre-definidas para el cuadro de búsqueda:

<input id="car" type="text" list="colores" />

<datalist id="colores">

<option value="Rojo">

<option value="Verde">

<option value="Amarillo">

</datalist>

<option> define las opciones en una lista desplegable para que el usuario pueda seleccionar.

El ID del elemento datalist debe coincidir con el atributo list del cuadro de entrada.

**Creando más campos**

Otros tipos de entrada nuevos incluyen email, url, y tel:

<input id="email" name="email" type="email"

placeholder="example@example.com" />

<input id="url" name="url" type="url"

placeholder="example.com" />

<input id="tel" name="tel" type="tel"

placeholder="555.555.1211" />

Estos campos son especialmente útiles cuando se abre una página en un dispositivo móvil moderno, que reconoce los tipos de entrada y abre el correspondiente teclado que coincide con el tipo de campo.

Estos nuevos tipos hacen más fácil estructurar y validar los formularios HTML.