

Capítulo 2

HTML5 y CSS3

Objetivo

Al finalizar el capítulo, el alumno:

- Aplicar las características de HTML 5.
- Implementar hojas de estilo haciendo uso de SASS.

Temas

1. HTML5
2. Estructuras y esquemas
3. Nuevos inputs, selectores, elementos
4. CSS3 con SASS
5. Canvas y SVG
6. Audio y video
7. Geolocalización


1. HTML5

HTML5

- HTML5 es la nueva versión del Lenguaje de Marcado de Hipertexto, y cambia los paradigmas de desarrollo y diseño web al introducir herramientas notables como etiquetas que permiten la publicación de archivos de audio y video con soportes de distintos códecs, etiquetas para contenidos en 2D y 3D, cambios en los formularios, y una web semántica mucho mejor aprovechada.
- Evolución de la web: <http://evolutionofweb.appspot.com>

7 - 4

Copyright © Todos los Derechos Reservados - Cibertec Perú S.A.C.



HTML5 es la nueva versión del Lenguaje de Marcado de Hipertexto, y cambia los paradigmas de desarrollo y diseño web.

Algunas características de HTML5:

- Simplificación: el nuevo código ofrece nuevas formas más sencillas de especificar algunos parámetros y piezas de código.
- Contenido multimedia: reproducción de audio y video sin necesidad de plugins o utilización de objetos embebidos en las páginas web.
- Animaciones: posibilidad de mostrar contenidos de manera similar a Flash, pero prescindiendo de este componente.
- Efectos y nueva versión de hojas de estilos: la nueva versión de HTML es acompañada de una nueva versión de hojas de estilo CSS, denominado CSS3. Se trata de nuevas posibilidades de formato, como por ejemplo, la implementación de sombras, bordes redondeados, etc.
- Muchas de las cosas que hasta ahora solo podrían lograrse insertándolas como imágenes, podrán realizarse con código. Esto no solo se traduce en una mejora de la velocidad y performance de un sitio, sino también en nuevas e ilimitadas opciones de diseño.
- Geolocalización: los sitios web podrán saber la ubicación física del usuario que lo visita.

- Tipografías no estándar: la mayor limitación a la que se enfrentaban los diseñadores era la imposibilidad de utilizar tipografías no estándar en sitios web. Prácticamente, todos estaban limitados a aquellas que fueron impuestas por los navegadores principales como: Arial, Times New Roman, Verdana, Tahoma, etc. Considerar que la implementación de sistemas como Google Fonts hoy, permite utilizar muchas más.

Por lo tanto, la nueva versión HTML5, contiene elementos dedicados ampliamente a mejorar la experiencia del usuario en una página web, haciendo más fácil poder agregar elementos de audio, video y en general de la web 2.0, así como, organizar contenidos utilizando menos código.

La nueva versión es más eficiente y ocupa menos recursos en el equipo del cliente; en particular, mediante el uso del nuevo reproductor que no requiere flash o Adobe Player para utilizarse, siendo HTML5 compatible con las versiones anteriores de HTML.

A pesar de que en el futuro se espera que HTML5 y CSS3 funcionen a la perfección en todos los navegadores, incluyendo los dispositivos móviles, por ahora se ejecuta mejor en Google Chrome, Safari y Mozilla Firefox, siendo Microsoft Internet Explorer el que se queda muy atrás en esta carrera (únicamente los navegadores 8.0 en adelante lo soportan, y muy pocas características).

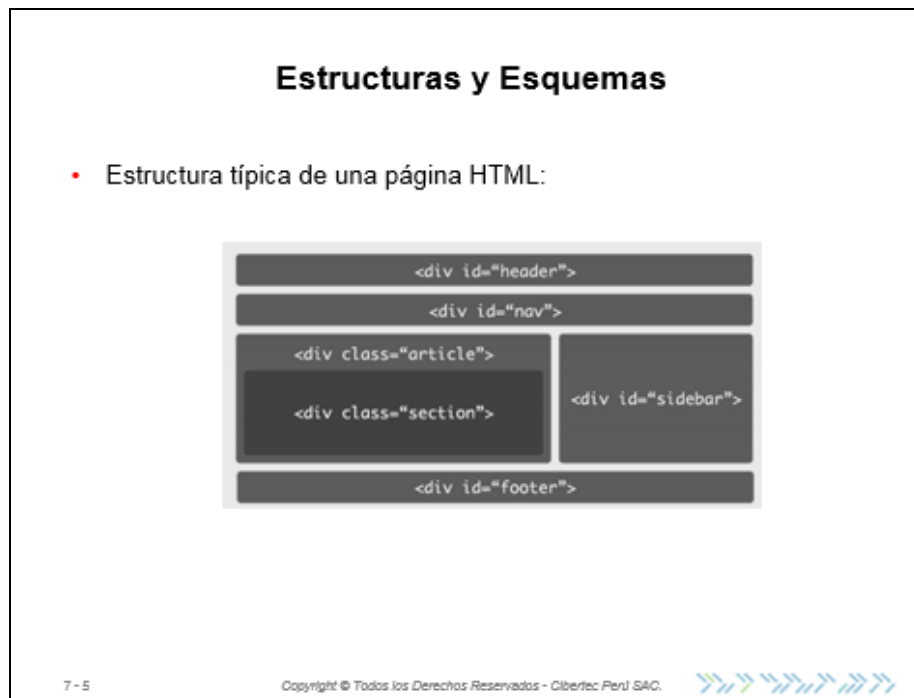
Entonces, dado que no todas las características que ofrecen HTML5 y CSS3 son admitidas por los navegadores web actuales, es necesario consultar algunas páginas de referencia para conocer qué elementos se pueden utilizar porque son admitidos por la mayoría de los navegadores y qué elementos no se pueden utilizar porque no funcionarán:

HTML5 Test: <http://html5test.com/>

Can I use: <http://caniuse.com/>

FindmebyIP: <http://fmbip.com/>

2. Estructuras y Esquemas



Los tags semánticos son aquellas etiquetas que dan un significado a las partes del documento, es decir, son etiquetas que indican cuál es el tipo de contenido que tienen. Hacer cosas como `<div id="header">` no tiene mucho sentido cuando el 99% de los proyectos web, tienen una cabecera, por eso con HTML5 tenemos `<header>`. Este es el propósito de los tags semánticos y todos siguen la misma idea.

Los principales tags semánticos de HTML5 no tienen una representación especial en pantalla. Todas se comportan como un `<div>` o un ``, pero cada una tiene un significado diferente:

- `<header>`: Representa la cabecera de una página.
- `<hgroup>`: Muchos headers necesitan múltiples títulos, como un blog que tiene un título y un tagline explicando el blog. De tal manera que `<hgroup>` permite colocar un `h1`, `h2` y `h3` dentro del header, sin afectar el SEO, permitiendo usar otro `h1` en el sitio. En el HTML actual, solo se puede usar `h1`, una vez por sitio o el `h1` pierde prioridad de SEO.
- `<nav>`: Está diseñado para colocar la navegación principal. Se puede colocar cualquier etiqueta dentro, aunque lo recomendado es usar listas ``.
- `<section>`: Define un área de contenido única dentro del sitio. En un blog, sería la zona donde están todos los posts. En un video de youtube, habría un section para el video, uno para los datos del video, otro para la zona de comentarios.
- `<article>`: Define zonas únicas de contenido independiente. En el home de un blog, cada post sería un article. En un post del blog, el post y cada uno de sus comentarios sería un `<article>`.

- `<aside>`: Cualquier contenido que no esté relacionado con el objetivo primario de la página va en un aside. En un blog, obviamente el aside es la barra lateral de información. En el home de un periódico, puede ser el área de indicadores económicos.
- `<footer>`: Representa el pie de página y todo lo que lo compone.

```
<header>
  <hgroup class="title">
    <h1>Titulo</h1>
  </hgroup>
</header>
<nav>Colocar Botones de Navegacion</nav>
<section>
  <article>
    <p>
      Informacion 1
    </p>
  </article>
  <article>
    <p>
      Informacion 2
    </p>
  </article>
</section>
<aside>
  <h3>Titulo Aside</h3>
  <p>
    Informacion 3
  </p>
  <ul>
    <li>Index</li>
    <li>About</li>
    <li>Contact</li>
  </ul>
</aside>
<footer>
  Informacion 4
</footer>
```

Se deben usar las etiquetas semánticas de HTML5 donde sean necesarias, y simplemente `div` cuando se necesite una división a la cual se le aplicará algún diseño gráfico o cualquier otra cosa que no tenga significado semántico.

3. Nuevos Input, Atributos, Elementos

Nuevos Inputs, Atributos, Elementos

- **Nuevos Inputs HTML5:**

Algunos muestran directamente algún contenido sobre el navegador y otros son para realizar validación. En caso que el navegador no soporte algún Input, simplemente se mostrará por defecto `<input type="text">`.

```
<input type="color" value="#b97a40">  
<input type="date" value="2013-01-01">  
<input type="datetime" value="2013-01-01T16:25:14.12">  
<input type="datetime-local" value="2013-01-01T15:41">  
<input type="email" value="ejemplo@cibertec.edu.pe">  
<input type="month" value="2013-01">  
<input type="number" value="6" min="0" max="50">  
<input type="range" min="0" max="50">  
<input type="search" value="[Buscar...]">  
<input type="tel" value="[Numero...]">  
<input type="time" value="15:28">  
<input type="url" value="http://www.cibertec.edu.pe">  
<input type="week" value="2013-W24">
```

7 - 9

Copyright © Todos los Derechos Reservados - Cibertec Perú SAC.



1. Nuevos Inputs

En muchas ocasiones, la entrada de datos puede convertirse en una tarea tediosa para los desarrolladores, quienes tienen que buscar qué métodos son los mejores para introducir datos, de una manera intuitiva y validarlos por medio de la programación.

Es frecuente ver, en aplicaciones web medianamente bien realizadas, rutinas de validación de datos que permiten al usuario, rellenar los formularios con la información correcta. El trabajo para la validación de los datos, generalmente, se realiza con un lenguaje del lado del cliente como Javascript.

Gracias a HTML5 los desarrolladores poseen importantes herramientas para validar datos en un formulario, de una forma más fácil, con menos rutinas de código.

HTML5 está pensado para ser usado en múltiples entornos, y por tal razón se han creado nuevos tipos de INPUT:

- **INPUT tel:** Este tipo de input viene predispuesto con un formato para escribir números telefónicos. En realidad no hace validación, pero sí se puede implementar una con la nueva API de validación de JavaScript.
- **INPUT number:** Sirve para escribir solo números. En algunos navegadores, cuando se ejecuta el evento `onSubmit`, no se hace el envío en caso que el campo number esté lleno de caracteres que no sean numéricos.

- INPUT search: Además de proporcionar un campo de entrada, se le agrega un ícono de búsqueda para distinguirlo de un campo de navegación.
- INPUT color: Este input brinda una paleta de colores donde el usuario puede escoger un color de forma dinámica. Es lo que, generalmente, se llama un colorpicker, con la particularidad que lo ofrece el propio navegador.
- INPUT range: Proporciona un control que se desliza, cambiando automáticamente el valor del campo.
- INPUT URL: Este tipo de entrada viene con un formato para URL absoluta.
- INPUT email: Tiene la capacidad de aceptar únicamente, direcciones de correo electrónico. Además, se pueden enviar varios emails, separados por comas, si se tiene especificado el atributo múltiple.
- INPUT datetime: Para obtener fecha del conjunto de la zona horaria UTC.
- INPUT date: Para introducir una fecha que no haga parte del conjunto horario.
- INPUT month: Para introducir meses del año.
- INPUT week: Ofrece una utilidad para escribir y captar información de semanas.
- INPUT time: Obtiene información con horas, minutos y segundos.

2. Nuevos Atributos

A continuación, se describen los atributos más importantes:

- Atributo autocomplete: La mayoría de los navegadores tienen algún tipo de funcionalidad de autocompletar. Este atributo permite controlar cómo funciona esto; especifica si un campo de formulario o de entrada, debe tener autocompletado de encendido o apagado, es decir, On u Off.
- Atributo autocomplete trabaja con <form> y con los <input> siguientes: text, search, url, tel, email, password, datepickers, range y color.
- Atributo autofocus: El autofocus o enfoque automático proporciona una forma declarativa para enfocar un control de formulario durante la carga de la página. Anteriormente, con JavaScript se usaba control.focus(). La nueva forma permite al navegador, hacer cosas inteligentes como no centrar el control, si el usuario ya está escribiendo en otro.
- Atributo múltiple: Especifica que el usuario puede introducir más de un valor en el elemento <input>; por ejemplo, <input type=email múltiples>, permite al usuario, introducir varias direcciones de correo electrónico.
- Atributo required: En un elemento se establecerá en un <input> y automáticamente, hace que el usuario se vea obligado a llenar el campo para continuar, es decir, el navegador no permitirá que se envíe la forma, sin que el input con este atributo, se encuentre vacío. Atributo requerido trabaja con los tipos de entrada: text, search, url, tel, email, password, date pickers, number, checkbox, radio y file.
- Atributo min and max: Especifica el valor mínimo y máximo para un elemento <input>. Funciona con los siguientes tipos de entrada: number, range, date, datetime, datetime-local, month, time y week.

3. Nuevos Elementos

A continuación, se describen los atributos más importantes:

- `<video>`: Inserta video sin necesidad de plugins.
- `<audio>`: Inserta audio sin necesidad de plugins.
- `<canvas>`: Se trata de un contenedor donde se permite el dibujo, el cual generalmente se realiza a través de JavaScript.
- `<svg>`: Inserta dibujos y animaciones vectoriales al estilo de Flash. Está basado en el estándar abierto SVG (Scalable Vector Graphics), derivado de XML.
- `<bdi>`: Permite aislar una parte del texto, de forma que este pueda ser formateado en una dirección diferente al resto del texto del documento. Es útil cuando se introduce texto como un nombre de usuario, por ejemplo, dentro de otro que puede variar su sentido de lectura (por ejemplo: árabe o idiomas orientales).
- `<command>`: Define un comando (puede ser un botón, un botón de selección múltiple, checkbox, radiobutton) que el usuario puede invocar. Un comando puede ser parte de un menú contextual, una barra de herramientas o formar parte de un menú usando el elemento `<menu>`, o puede ponerse en cualquier parte de la página para definir un atajo de teclado.
- `<datalist>`: Especifica una lista de opciones predefinidas para un elemento `<input>`. Se usa para proporcionar la característica de autocompletado a los elementos de este tipo. Los usuarios verán una lista, que se va rellenando de forma automática, conforme ellos escriben sugiriendo diferentes opciones. La idea de esta etiqueta es normalizar las inclusiones de contenidos de aplicaciones externas vía plugins.
- `<meter>`: Define una medida escalar, dentro de un rango conocido.
- `<progress>`: Representa el grado de progreso de una tarea.


4. CSS3

CSS3

- CSS = Cascading Style Sheets (hojas de estilo en cascada)
- Mientras que HTML nos permite definir la estructura de una página web, los CSS nos ofrece la posibilidad de definir las reglas y estilos de representación en diferentes dispositivos capaces de mostrar contenidos web.
- En la versión CSS3, se realiza la incorporación de nuevos mecanismos para mantener un mayor control sobre el estilo de los elementos, sin tener que recurrir a trucos o hacks.

7 - 13

Copyright © Todos los Derechos Reservados - Cibertec Perú SAC.



Las hojas de estilo en cascada proporcionan un medio para aplicar cambios a la presentación de una estructura HTML, definiendo cómo se muestran los elementos. Mediante el uso de CSS, puede definir colores de fondo, márgenes, bordes, fuentes, posiciones, y mucho más, es decir, se tiene el control de la apariencia de los elementos HTML.

Para aprender cómo podemos crear nuestra propia organización de los elementos en pantalla, debemos primero entender cómo los navegadores procesan el código HTML. Los navegadores consideran cada elemento HTML como una caja (por ejemplo: div o span). Una página web es en realidad un grupo de cajas ordenadas siguiendo ciertas reglas. Estas reglas son establecidas por estilos provistos por los navegadores o por los diseñadores usando CSS.

CSS tiene un set predeterminado de propiedades destinados a sobrescribir los estilos provistos por navegadores y obtener la organización deseada. Estas propiedades no son específicas, tienen que ser combinadas para formar reglas que luego serán usadas para agrupar cajas y obtener la correcta disposición en pantalla. La combinación de estas reglas es normalmente llamada modelo o sistema de disposición. Todas estas reglas aplicadas juntas constituyen lo que se llama un modelo de caja.

El objetivo de las nuevas características de CSS3, es evitar los hacks (o trucos) en programación que se debían hacer a la hora de crear diseños cotidianos. Generalmente, estos hacks eran ingeniosos, pero requerían mucho código (por lo general poco mantenible), lo cual hacía el código de las páginas más complicadas.

Estas son algunas de las características que varían en CSS3:

- Bordes
 - border-color: se añaden bordes con degradados (gradientes).
 - border-image: posibilidad de crear bordes redondos y de muchas formas para imágenes.
 - border-radius: permite hacer cajas con bordes redondeados.
 - box-shadow: sombras con gradientes para las cajas.
 - Fondos
 - background-origin & background-clip: posiciones del fondo con respecto a las cajas, con tres modos diferentes.
 - background-size: especificación del tamaño de la imagen de fondo, en píxeles o porcentaje.
 - multiple backgrounds: varias imágenes de fondo en el mismo elemento. Su notación es como hasta ahora, pero separando con comas cada imagen.
 - Color
 - Colores HSL: otra manera de especificar los colores: color, saturación y luminosidad.
 - Colores HSLA: lo mismo que el anterior, añadiendo un canal alfa (opacidad).
 - Colores RGBA: añade un cuarto canal a la notación RGB de colores, el canal alfa (opacidad).
 - Opacidad: lo comentado en los dos puntos anteriores.
 - Texto
 - text-shadow: efectos de sombra en texto.
 - text-overflow: cuando el texto desborde, se ponen automáticamente tres puntos suspensivos.
 - text-wrap: trunca palabras largas que no caben en una línea.
 - Layout
 - box-sizing: nuevo atributo para especificar las dimensiones.
 - resize: para redimensionar las cajas.
 - outline: para crear marcos y bordes dobles con la separación especificada en píxeles.
 - Otros
 - media queries: Adapta la prestación de los documentos HTML basado en condiciones tales como la resolución de pantalla y la orientación, para dar cabida a diferentes dispositivos como teléfonos inteligentes y tabletas.
 - multi-columnlayout: propiedades de columnas de texto.
 - Web fonts: se implementó en CSS2, esta propiedad permite que se visualice correctamente un texto con una fuente que el usuario no tenga instalada en su sistema operativo.
 - Speech: se implementó en CSS2, pero ahora se le añaden (y quitan) muchos atributos que sirve para que una página pueda ser leída por la computadora en “voz alta”. Útil para que la web no suponga una barrera para usuarios con problemas de visión.
- Para todas las características, ir a: <http://www.w3schools.com/css>

5. Canvas y SVG

Canvas y SVG

- **Canvas**
Permite representar textos, imágenes, gráficos y formas geométricas de todo tipo, actualizando su contenido en tiempo real. Para ello, utiliza el API Canvas 2D y la aceleración de gráficos por hardware.

```
<canvas id="miCanvas" width="200" height="100">
  Tu navegador no soporta Canvas
</canvas>

<script>
  var c = document.getElementById("miCanvas");
  var ctx = c.getContext("2d");

  // Crear Gradiente
  var grd = ctx.createRadialGradient(75, 50, 5, 90, 60, 100);
  grd.addColorStop(0, "blue");
  grd.addColorStop(1, "white");

  // Rellenar con Gradiente
  ctx.fillStyle = grd;
  ctx.fillRect(10, 10, 150, 80);
</script>
```

7 - 16 Copyright © Todos los Derechos Reservados - Cibertec Perú SAC.

1. Canvas

El elemento Canvas permite especificar un área de la página donde se puede, a través de scripts, dibujar y renderizar imágenes, lo que amplía notablemente las posibilidades de las páginas dinámicas y permite hacer cosas que hasta ahora estaban reservadas a los desarrolladores en Flash. La ventaja es que para usar Canvas, no es necesario ningún “plugin” en el navegador.

Uno de los problemas de Canvas es que se creó bajo propiedad intelectual de Apple, es decir, que dicha empresa era la creadora de la ingeniería que daba soporte a este nuevo elemento, y por tanto, se encontraba bajo patentes de la compañía. Este hecho, añadido a la existencia de un formato abierto que sirve para hacer cosas similares como es el SVG, hizo que surgiera una polémica sobre la aceptación de este elemento en el nuevo estándar del HTML5. No obstante, Apple abrió la licencia de uso de patente, liberando la propiedad intelectual de la misma, condición estrictamente necesaria para que la W3C, que siempre apoya patentes libres, incorporase, finalmente, dentro del nuevo estándar del lenguaje HTML5.

Canvas permite dibujar en la página y actualizar dinámicamente estos dibujos por medio de scripts y atendiendo a las acciones del usuario. Todo esto da unas posibilidades de uso tan grandes como las que se disponen con el plugin de Flash, en lo que respecta a renderización de contenidos dinámicos. Las aplicaciones pueden ser grandes, desde juegos, efectos dinámicos en interfaces de usuario, editores de código, editores gráficos, aplicaciones, efectos 3D, etc.

Cuando se trabaja con Canvas, primero se debe referenciar el elemento Canvas y adquirir su contexto (API). Por el momento, el único contexto disponible es el contexto bidimensional.

Una vez que se adquiere el contexto, se puede empezar a dibujar en la superficie del canvas usando la API. La API bidimensional ofrece muchas de las herramientas que podemos encontrar en cualquier aplicación de diseño gráfico como Adobe Illustrator o Inkscape: trazos, rellenos, gradientes, sombras, formas y curvas Bézier. Pero de todas formas, se debe especificar cada acción usando JavaScript.

Dibujando con javascript:

- Se recomienda usar firebug o similares a la hora de aprender a usar la API para poder ver los cambios en vivo y avanzar con mayor velocidad.
- Referenciar al elemento Canvas:

```
<canvas id="canvasDAT" width="360" height="240">
  <p>Tu navegador no soporta canvas!!!</p>
</canvas>

var canvas = document.getElementById('canvasDAT');
var contexto = canvas.getContext('2d');
```

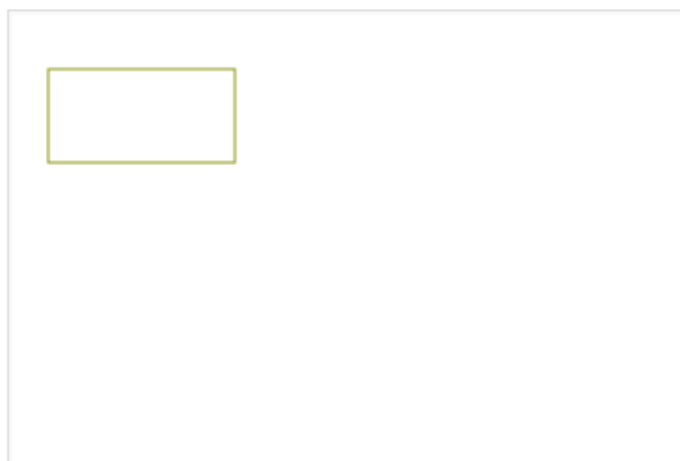
- Especificar el color del trazo: (en el ejemplo, este color es verde)

```
contexto.strokeStyle = '#8f9814';
```

- Especificar la figura a dibujar:

```
contexto.strokeRect(20, 30, 100, 50);
```

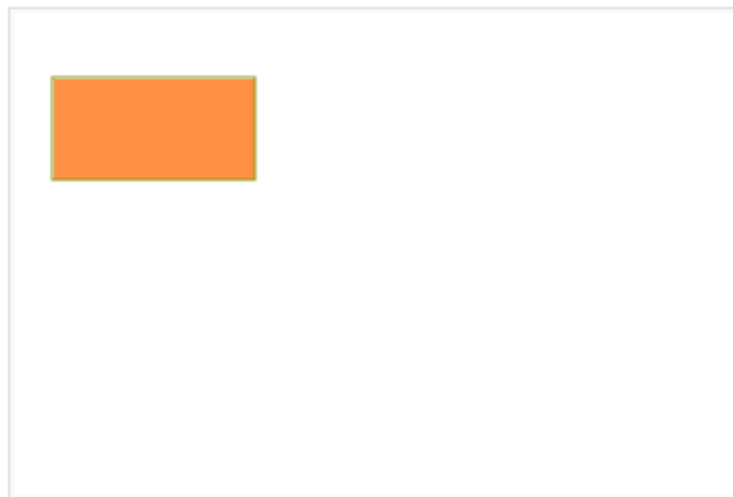
La sintaxis del método strokeRect es como sigue: strokeRect (izquierda, arriba, ancho, largo). Por lo tanto, en el ejemplo, se ha dibujado un rectángulo posicionado a 20 pixels del margen izquierdo y 30 del derecho con un tamaño de 100x50 pixels.



- Si se desea dibujar un rectángulo relleno con color, se deberá además especificar el color del relleno, tal como sigue:

```
contexto.fillStyle = '#ff8f43';
```

```
contexto.fillRect(20, 30, 100, 50);  
contexto.strokeRect(19, 29, 101, 51);
```



La API 2D tiene muchos métodos para trabajar con ella. En teoría, se puede crear cualquier imagen que puede dibujarse en un programa de edición de gráficos vectoriales.

El potencial de Canvas reside en su habilidad para actualizar su contenido en tiempo real. Si se usa esa habilidad para responder a eventos de usuario, se pueden crear herramientas y juegos que, anteriormente a la nueva especificación, hubiesen requerido de un plugin externo como Flash.

2. SVG (Scalable Vector Graphics)

La renderización en el Canvas se hace como un mapa de bits, lo que significa que los píxeles a mostrar están ordenados, y después de la forma mostrada, el Canvas se queda con los resultados de mapa de bits. En otras palabras, el Canvas no almacena el hecho de que ha creado un rectángulo. Si la superficie de dibujo se escala más grande, el Canvas solo tiene los píxeles para trabajar, por lo que el resultado es una versión pixelada del gráfico.

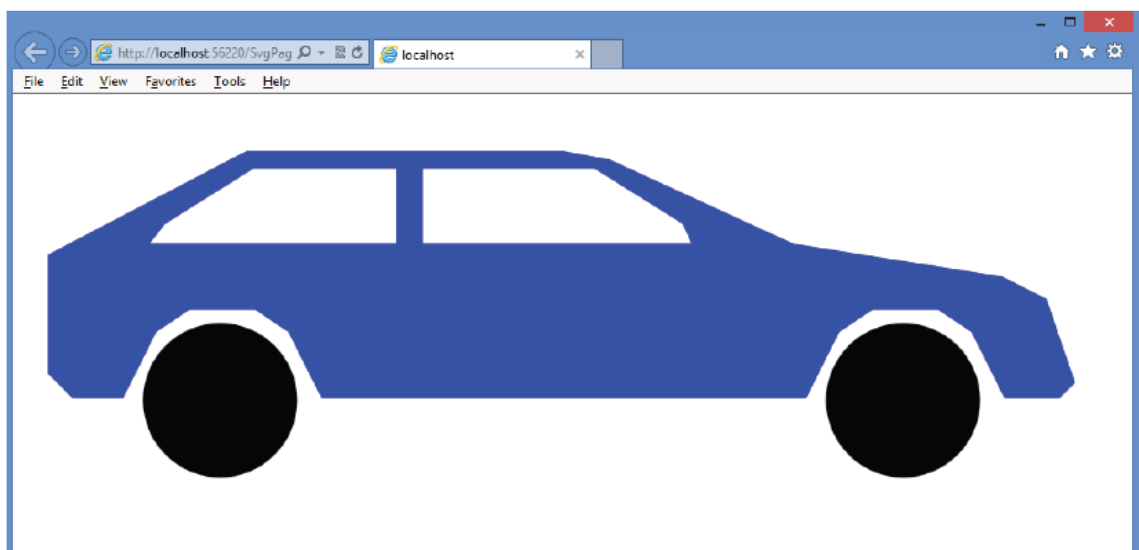
SVG es diferente, los comandos se almacenan en una manera que permite ser re-ejecutado. Si el tamaño de la superficie de dibujo cambia, los comandos se pueden escalar y volver a ejecutar para el nuevo dibujo y crear una nueva imagen. El resultado es que se ve una imagen nítida independientemente de la escala.

Aunque SVG se renderiza y escala mucho mejor que el Canvas, se necesita tiempo para escalar y volver a ejecutar los comandos, por lo que el rendimiento

no es tan bueno. El Canvas es preferible cuando el rendimiento es prioridad sobre la apariencia al cambiar de escala un dibujo.

A continuación, se muestra un ejemplo de cómo crear un dibujo con SVG, el cual puede ser redimensionado sin perder resolución:

```
<svg width="100%" height="100%" xmlns="http://www.w3.org/2000/svg" viewBox="0 50 500 175">
  <path d="m267 76 l-21 -4 -144 0 -90 47 0 54 11 11 23 0 15 -30 15 -10 30 0
    15 10 15 30 220 0 15 -30 15 -10 30 0 15 10 15 30 125 0 7 -7
    -13 -38 -20 -10 -95 -15 z" fill="blue" id="carroceria" />
  <path d="m65 105 l40 -25 65 0 0 34 -112 0 z" fill="white" id="ventanaTrasera" />
  <path d="m300 105 l-40 -25 -78 0 0 34 122 0 z" fill="white" id="ventanaDelantera" />
  <circle r="35" cy="185" cx="90" fill="black" id="ruedaTrasera" />
  <circle r="35" cy="185" cx="400" fill="black" id="ruedaDelantera" />
</svg>
```



El elemento `<path>` es una secuencia de comandos que crean una forma compleja definiendo un camino a dibujar. Los atributos del elemento path son:

- id: identificador del path.
- fill: el color con el que desea llenar el path.
- d: recibe comandos y posiciones, por lo general comienza con un comando para pasar al punto en el que comienza el dibujo de la forma.

Algunos de los comandos posibles: (cuando el comando es en mayúsculas se refiere a coordenadas absolutas, en minúsculas es para coordenadas relativas).

- M o m: moverse a una coordenada específica.
- L o l: Dibuja una línea desde la posición actual a la coordenada especificada.
- H o h: Dibuja una línea horizontal desde la posición actual hasta el nuevo valor x especificado en el mismo plano horizontal.
- V o v: Dibuja una línea vertical desde la posición actual hasta el nuevo valor y especificado en el mismo plano vertical.

- Z o z: cierra la ruta desde la posición actual hasta el principio del camino.

El elemento `<circle>`, tiene los atributos `r`, `cx`, `cy`, `relleno` e `id`. El atributo `r` establece el radio del círculo. Los atributos `cx` y `cy` marcan el centro del círculo de coordenadas. El atributo de `relleno` define el color del círculo.

El atributo `viewBox` describe la parte del área de dibujo que desea que se vea. A pesar de que el dibujo cubre toda la pantalla del navegador, la figura solo puede existir en una pequeña parte del dibujo. El atributo `viewBox` permite indicar un zoom en la parte destinada a la figura y eliminar el espacio en blanco adicional. Es decir, se usa `viewBox` para obtener las capacidades de zoom adecuadas al cambiar el tamaño de la página HTML. Tiene cuatro parámetros: la coordenada X mínima, la coordenada Y mínima, el ancho y la altura. Esto permite describir el área rectangular que se mostrará.

6. Audio y Video

Audio y Video

- **Audio**

```
<audio src="audiodat.mp3" controls>
</audio>
```

```
<audio controls>
  <source src="audiodat.ogg" type="audio/ogg" />
  <source src="audiodat.mp3" type="audio/mpeg" />
  <object type="application/x-shockwave-flash" data="player.swf?soundFile=audiodat.mp3">
    <param name="movie" value="player.swf?soundFile=audiodat.mp3" />
  </object>
</audio>
```

7 - 19
Copyright © Todos los Derechos Reservados - Cibertec Perú S.A.C.

Con el aumento del ancho de banda, los contenidos multimedia han ido aumentando de forma vertiginosa hasta convertirse en una de las mayores necesidades de ancho de banda en la red. El método por excelencia, a la hora de reproducir recursos multimedia es a través de un navegador, y desde sus inicios se hizo utilizando plugins.

El problema con utilizar un reproductor de audio o video basado en plugins, es que su contenido está encerrado y oculto para el resto del documento. Disponer de elementos nativos en HTML, supone la integración de los mismos con otras tecnologías del navegador como JavaScript y CSS.

1. Audio

Este elemento permite el uso de diferentes formatos de archivo. Los formatos que soportan los diferentes navegadores no son parte del estándar, sino que depende de la implementación de cada fabricante. Según la siguiente tabla, se puede ver qué formatos soportan los navegadores más usados de forma nativa:

Códec	Tipo	IE >=9	Firefox	Chrome	Safari	Opera
Ogg Vorbis	Libre (BSD)	no	si	si	no	si
WAV PCM	Privativo (Microsoft, IBM)	no	si	si	si	si
MP3	Privativo (En disputa)	si	no	si	si	si en Linux y FreeBSD
AAC	Bajo patente AAC	si	no	si	si	si en Linux y FreeBSD
Speex	Libre (BSD)	no	no	si	no	no

La forma básica es embeber un archivo de audio de la siguiente manera:

```
<audio src="audiodat.mp3">
</audio>
```

- Para que el archivo de audio se autoreproduzca al cargar la página, se usa el atributo `autoplay`:

```
<audio src="audiodat.mp3" autoplay>
</audio>
```

- Si se desea que se repita indefinidamente, se usa el atributo `loop`:

```
<audio src="audiodat.mp3" autoplay loop>
</audio>
```

- Se pueden cargar los controles básicos para el elemento audio, con el atributo `controls`. Este atributo hace que el navegador provea de una interfaz con controles para la reproducción y el control del volumen de forma nativa.

```
<audio src="audiodat.mp3" controls>
</audio>
```

Los controles del navegador pueden ser suplantados por los propios controles usando JavaScript, a través de una API que proporciona métodos para controlar todos los aspectos de la reproducción del archivo de audio.

```
<audio id="player" src="audiodat.mp3">
</audio>
<div>
  <button onclick="document.getElementById('player').play();">Reproducir</button>
  <button onclick="document.getElementById('player').pause();">Pausa</button>
  <button onclick="document.getElementById('player').volume += 0.1;">Subir Volumen</button>
  <button onclick="document.getElementById('player').volume -= 0.1;">Bajar Volumen</button>
</div>
```

- Si se desea que el archivo de audio sea precargado, en segundo plano por el navegador, entonces se puede usar el atributo `preload`, el cual puede tomar tres posibles valores: `none`, `auto` y `metadata`.

Safari precarga los archivos de audio por defecto, usando `preload="none"`, se puede asegurar de que eso no ocurra en aquellos contextos en los que se considere que no es necesario que el navegador precargue los archivos, por ejemplo, cuando existen muchos archivos en una misma página. Si solo se tiene un archivo de audio en la página, entonces, se podrá usar `preload="auto"`.

```
<audio src="audiodat.mp3" preload="none">
</audio>
```

- Existe una forma de definir más de un archivo de audio en diferentes formatos, utilizando únicamente una etiqueta audio para ello. En lugar de usar el atributo src en la etiqueta de apertura, se puede utilizar la etiqueta source para poder definir múltiples archivos.

```
<audio controls>
  <source src="audiodat.ogg" type="audio/ogg" />
  <source src="audiodat.mp3" type="audio/mpeg" />
</audio>
```

- Para cubrir el caso de navegadores que no soporten la funcionalidad de audio, se puede hacer lo siguiente: (añadir el plugin de Flash, a través de la etiqueta object)

```
<audio controls>
  <source src="audiodat.ogg" type="audio/ogg" />
  <source src="audiodat.mp3" type="audio/mpeg" />
  <object type="application/x-shockwave-flash" data="player.swf?soundFile=audiodat.mp3">
    <param name="movie" value="player.swf?soundFile=audiodat.mp3" />
  </object>
</audio>
```

2. Video

Existen diversos formatos de video para la nueva especificación, y por supuesto, no todos reproducen los mismos formatos de forma nativa:

Códec	Tipo	IE >=9	Firefox	Chrome	Safari	Opera
Ogg Theora	Libre	no ^[1]	si	si	no ^[2]	si
H.264	Propietario	si	no ^[3]	no	si	no
VP8	Libre	no ^[4]	si	si	no	si

El elemento video es muy parecido al elemento audio, también dispone de los atributos autoplay, loop y preload. También se puede especificar la fuente de un archivo usando el atributo src en la etiqueta de apertura, o bien usando el elemento source entre las etiquetas de apertura y cierre. Además, se pueden utilizar los controles que ofrece el navegador de forma nativa utilizando el atributo controls, o bien implementar controles propios en JavaScript.

Adicionalmente, el elemento video ocupa espacio en la ventana, por lo tanto, se puede definir un tamaño para el mismo.

```
<video src="videodat.mp4" controls width="360" height="240">
</video>
```

También, se puede definir una imagen representativa para el video para que sea mostrada al navegador como portada del elemento antes de la reproducción, usando el atributo póster:

```
<video src="videodat.mp4" controls width="360" height="240" poster="portadadat.jpg">
</video>
```

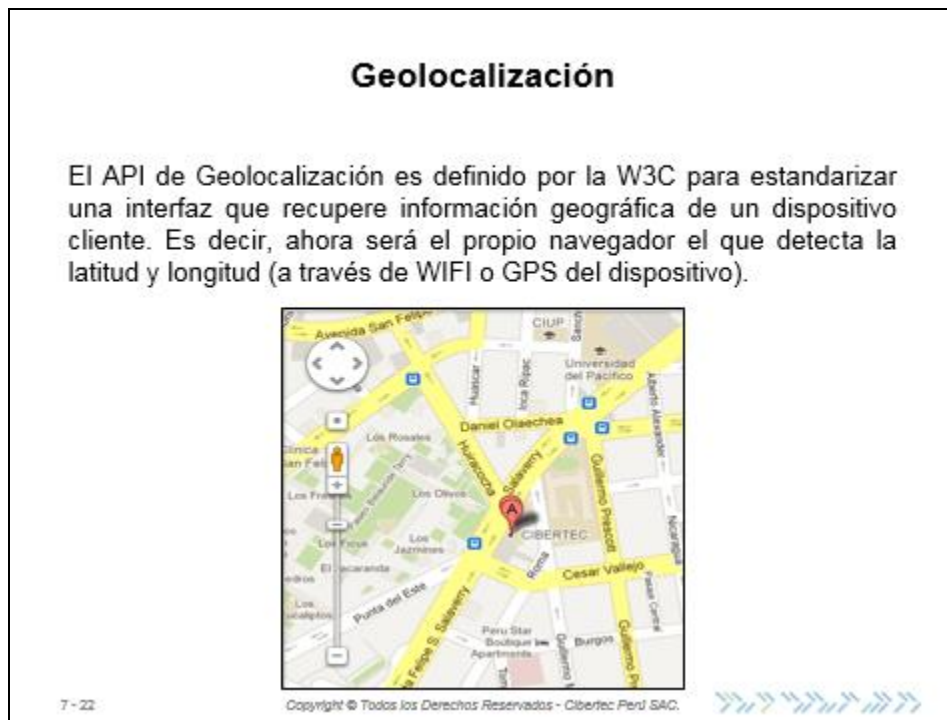
Al igual que con audio, se pueden especificar diferentes archivos en diferentes formatos para que todos puedan reproducir el contenido de video:

```
<video controls width="360" height="240" poster="portadadat.jpg">
  <source src="videodat.ogv" type="video/ogg" />
  <source src="videodat.mp4" type="video/mp4" />
</video>
```

Para cubrir el caso de navegadores que no soporten la funcionalidad de video, se puede hacer lo siguiente: (añadir el plugin de Flash, a través de la etiqueta object).

```
<video controls width="360" height="240" poster="poster.jpg">
  <source src="videodat.ogv" type="video/ogg" />
  <source src="videodat.mp4" type="video/mp4" />
  <object type="application/x-shockwave-flash" width="360" height="240" data="player.swf?file=videodat.mp4">
    <param name="movie" value="player.swf?file=videodat.mp4" />
  </object>
</video>
```

7. Geolocalización



La Geolocalización consiste en obtener la ubicación exacta del lugar en el mundo donde se encuentre el usuario, y opcionalmente, compartir esa información. Para esto, existe una API de geolocalización, la cual se trata de una interface de alto nivel para acceso a la información de localización, asociada solamente con el dispositivo host de la aplicación, tales como latitud y longitud.

La propia API es independiente de las fuentes de información de la ubicación. Las fuentes comunes de ubicación incluyen el Sistema de Posicionamiento Global (GPS = Global Positioning System) y la ubicación obtenida de señales de la red.

La API está diseñada para permitir obtener una posición por petición y/o peticiones repetidas para actualizar la posición, así como, la posibilidad de consultar las posiciones almacenadas en caché. La información de ubicación está representada por coordenadas de latitud y longitud, y se mostrará en un mapa informativo de Google.

A continuación, se muestra un código que obtiene varias peticiones de posiciones:

```
function scrollMap(position) {  
    // Scrolls the map so that it is centered at (position.coords.latitude, position.coords.longitude).  
}  
  
// Request repeated updates.  
var watchId = navigator.geolocation.watchPosition(scrollMap);  
  
function buttonClickHandler() {  
    // Cancel the updates when the user clicks a button.  
    navigator.geolocation.clearWatch(watchId);  
}
```

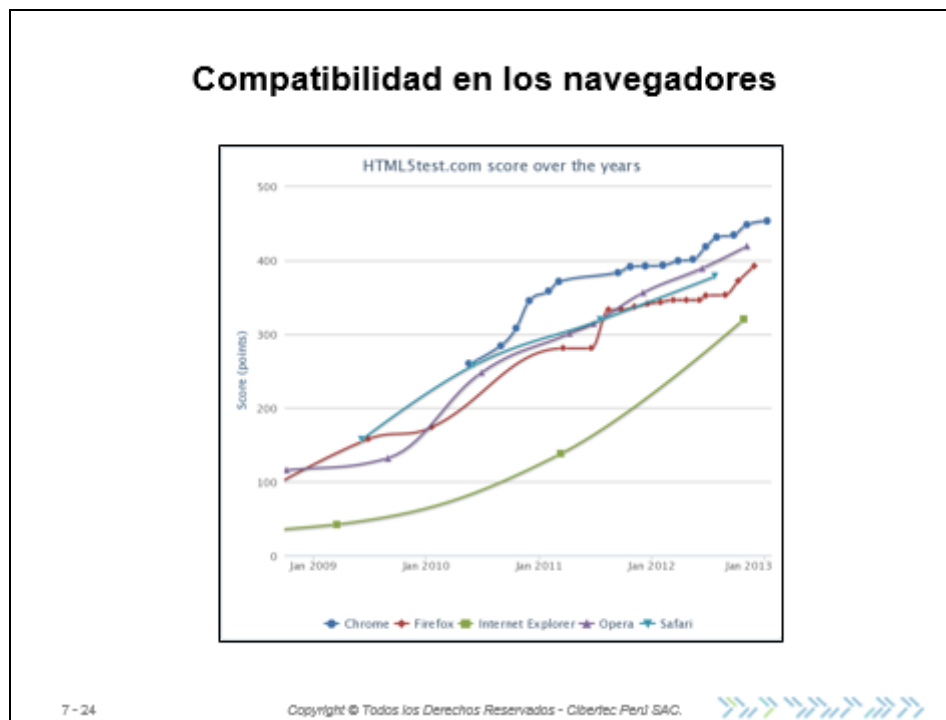
En este ejemplo, se tienen tres simples funciones usando JavaScript que ayudarán a visualizar la ubicación. La primera de ellas consiste en obtener la información de la ubicación y en definir si el navegador soporta esta nueva API.

```
if (navigator.geolocation) {  
    navigator.geolocation.getCurrentPosition(success, error);  
} else {  
    error('not supported');  
}
```

Si el navegador soporta esta funcionalidad, entonces se podrá obtener la información de la ubicación actual, es decir, será una sola petición. En caso de no lograr obtener la ubicación o el navegador no sea soportado, se mostrará un mensaje de error al obtener los datos, y si se logra obtener la información de la ubicación, esta se muestra en el mapa.

```
var map = new google.maps.Map(document.getElementById("mapcanvas"), myOptions);  
  
var marker = new google.maps.Marker({  
    position: latlng,  
    map: map,  
    title:"Estas Aqui! (" +position.coords.accuracy+" )"  
});
```

8. Compatibilidad en los navegadores

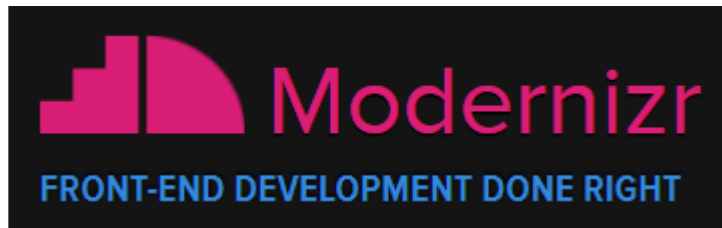


Las funcionalidades de HTML 5 y CSS3 tienen muchas características interesantes e importantes. Pero existe una brecha importante a superar, la compatibilidad o soporte HTML 5 que tienen los navegadores. El mayor problema viene con Internet Explorer (IE6, IE7, IE8), aunque desde IE9 en adelante ya tienen un funcionamiento aceptable. Con navegadores como Chrome y Firefox es un poco mejor, sobre todo con Chrome que está en evolución constante de acuerdo a los avances de HTML 5.

Es en este escenario que necesitamos de una librería javascript que permita detectar si el navegador es compatible con ciertas funcionalidades HTML 5 y CSS3. Esta librería es Modernizr.

Con ella, se puede desarrollar utilizando lo último del estándar, y detectar si el navegador empleado por el usuario es compatible. Con ello, si hubiese alguna funcionalidad que sea vital para la aplicación que el navegador no soporta, se puede ejecutar un código alternativo, por ejemplo, con jQuery, para que realice esa misma funcionalidad.

Los proyectos de ASP.NET MVC ya traen incluido dentro los scripts que agrega por default el Visual Studio. Pero también se puede agregar manualmente, descargándolo de la página oficial: <http://modernizr.com>



Se puede usar de dos maneras:

- Al referenciar a esta librería, contamos con un objeto llamado "Modernizr" en la página, el cual puede ser consultado usando javascript para tomar una acción dependiendo si la característica no es soportada. En el siguiente ejemplo, se consulta si el navegador soporta la característica de bordes redondeados:

```
if (!Modernizr.borderradius) {  
    //Logica para agregar bordes redondeados en browsers antiguos.  
}
```

- El mismo trabajo que en ejemplo anterior se puede hacer pero usando estilos de la siguiente manera:

```
.no-borderradius p {  
    /*Logica para agregar bordes redondeados en browsers antiguos.*/  
}
```

Para evitar el uso de hacks o implementar soluciones poco recomendadas como generar una serie de capas para simular una sombra o un borde redondeado, es que podemos apoyarnos de otras librerías, como por ejemplo: (solo es necesario descargarlas de sus páginas y agregarlas al proyecto)

- IE-CSS3: <http://fetchak.com/ie-css3>
A través de DirectX y VML, este script permite usar cosas como bordes redondeados y sombras sobre objetos de CSS3 en IE6, 7 y 8.

```
.no-borderradius p {  
    behavior: url(ie-css3.htc);  
}
```

- CSS3 PIE: <http://css3pie.com>
Hace a IE 6-9 capaz de mostrar varias de las características más útiles de decoración de CSS3.

```
.no-borderradius p {  
    behavior: url(PIE.htc);  
}
```