# Manual de Canvas del HTML 5

*Manual del elemento Canvas del HTML 5 en el que aprenderemos a dibujar sobre los lienzos, utilizando Javascript y el API de Canvas.*

# Parte 1:

# Introducción a HTML 5

Qué es HTML 5 y qué novedades traerá consigo en el desarrollo de páginas web, entre las que destaca el elemento CANVAS, que estamos explicando en el presente manual.

## 1.1.- Qué es HTML 5

*Veremos qué es HTML 5, su previsión de tiempo para convertirse en una especificación recomendada y las novedades más significativas que proporcionará.*

Ahora convendría explicar qué es exactamente HTML 5, ya que no es simplemente una nueva versión del lenguaje de marcación HTML, sino una agrupación de diversas especificaciones concernientes al desarrollo web. Es decir, HTML 5 no se limita sólo a crear nuevas etiquetas, atributos y eliminar aquellas marcas que están en desuso o se utilizan inadecuadamente, sino que va mucho más allá.

Así pues, HTML 5 es una nueva versión de diversas especificaciones, entre las que se encuentran:

• HTML 4

• XHTML 1

• DOM Nivel 2 (DOM = Document Objetc Model)

A la par, HTML 5 pretende proporcionar una plataforma con la que desarrollar aplicaciones web más parecidas a las aplicaciones de escritorio, donde su ejecución dentro de un navegador no implique falta de recursos o facilidades para resolver las necesidades reales de los desarrolladores. Para ello se están creando unas APIs que permitan trabajar con cualquiera de los elementos de la página y realizar acciones que hasta hoy era necesario realizar por medio de tecnologías accesorias.

Estas API, que tendrán que ser implementadas por los distintos navegadores del mercado, se están documentando con minuciosidad, para que todos los Browsers, creados por cualquier compañía las soporten tal cual se han diseñado. Esto se hace con la intención que no ocurra lo que viene sucediendo en el pasado, que cada navegador hace la guerra por su parte y los que acaban pagándolo son los desarrolladores y a la postre los usuarios, que tienen muchas posibilidades de acceder a webs que no son compatibles con su navegador preferido.

### 1.1.1.- Cuándo estará listo HTML 5

Según informan en la página de la organización WHATWG, HTML 5 se prevé esté listo como especificación de implementación recomendada en el 2012. ¿Quiere esto decir que vamos a tener que esperar hasta 2012 para aprovechar las ventajas de HTML 5? realmente no es justamente así, puesto que algunos navegadores ya implementan muchas de las características del moderno lenguaje.

Resulta que HTML 5 está formado por muchos módulos distintos, cuyo grado de especificación está en niveles dispares. Por tanto, muchas de las características de HTML 5 están ya listas para ser implementadas, en un punto de desarrollo que se encuentra cercano al que finalmente será presentado. Otras muchas características están todavía simplemente en el tintero, a modo de ideas o borradores iniciales.

De hecho, las versiones más nuevas de casi todos los navegadores, incluido el polémico Internet Explorer 8, implementan algunas de las características de HTML 5. Claro que, para que una web se vea bien en todos los sistemas, hay que utilizar sólo aquellas partes que funcionan en todos los navegadores, por lo que a día de hoy, pocas son las utilidades realmente disponibles del lenguaje, si queremos hacer un sitio web compatible. No obstante, en el peor de los casos, podemos empezar a usar a nivel experimental estas características, aunque

sólo sea para frotarnos las manos en espera de incorporarlas realmente en nuestras prácticas de desarrollo habituales.

### 1.1.2.- Cuáles son las novedades de HTML 5

HTML 5 incluye novedades significativas en diversos ámbitos. Como decíamos, no sólo se trata de incorporar nuevas etiquetas o eliminar otras, sino que supone mejoras en áreas que hasta ahora quedaban fuera del lenguaje y para las que se necesitaba utilizar otras tecnologías.

• **Estructura del cuerpo**: La mayoría de las webs tienen un formato común, formado por elementos como cabecera, pie, navegadores, etc. HTML 5 permite agrupar todas estas partes de una web en nuevas etiquetas que representarán cada uno de las partes típicas de una página.

• **Etiquetas para contenido específico**: Hasta ahora se utilizaba una única etiqueta para incorporar diversos tipos de contenido enriquecido, como animaciones Flash o vídeo. Ahora se utilizarán etiquetas específicas para cada tipo de contenido en particular, como audio, vídeo, etc.

• **Canvas**: es un nuevo componente que permitirá dibujar, por medio de las funciones de un API, en la página todo tipo de formas, que podrán estar animadas y responder a interacción del usuario. Es algo así como las posibilidades que nos ofrece Flash, pero dentro de la especificación del HTML y sin la necesidad de tener instalado ningún plugin.

• **Bases de datos locales**: el navegador permitirá el uso de una base de datos local, con la que se podrá trabajar en una página web por medio del cliente y a través de un API. Es algo así como las Cookies, pero pensadas para almacenar grandes cantidades de información, lo que permitirá la creación de aplicaciones web que funcionen sin necesidad de estar conectados a Internet.

• **Web Workers**: son procesos que requieren bastante tiempo de procesamiento por parte del navegador, pero que se podrán realizar en un segundo plano, para que el usuario no tenga que esperar que se terminen para empezar a usar la página. Para ello se dispondrá también de un API para el trabajo con los Web Workers.

• **Aplicaciones web Offline**: Existirá otro API para el trabajo con aplicaciones web, que se podrán desarrollar de modo que funcionen también en local y sin estar conectados a Internet.

• **Geolocalización**: Las páginas web se podrán localizar geográficamente por medio de un API que permita la Geolocalización.

• **Nuevas APIs para interfaz de usuario**: temas tan utilizados como el "drag & drop" (arrastrar y soltar) en las interfaces de usuario de los programas convencionales, serán incorporadas al HTML 5 por medio de un API.

• **Fin de las etiquetas de presentación**: todas las etiquetas que tienen que ver con la presentación del documento, es decir, que modifican estilos de la página, serán eliminadas. La responsabilidad de definir el aspecto de una web correrá a cargo únicamente de CSS.

Como se puede ver, existirán varios API con los que podremos trabajar para el desarrollo de todo tipo de aplicaciones complejas, que funcionarán online y offline. Quizás se entienda mejor por qué HTML 5 es un proyecto tan ambicioso y que está llevando tanto tiempo para ser elaborado.

## 1.2.- Introducción a Canvas del HTML 5

*Canvas es uno de los componentes más novedosos de estándar HTML 5 que sirve para dibujar dinámicamente imágenes en una página web.*

HTML 5 viene con varias novedades interesantes y una de las que podemos empezar a probar ya es el Canvas. Canvas significa en español algo así como lienzo y es básicamente eso, un área donde podemos dibujar como si fuera un lienzo.

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, con la ventaja que para usar canvas no será necesario ningún plugin en el navegador, lo que mejorará la disponibilidad de esta nueva aplicación.

En este artículo y los siguientes pretendemos dar una introducción a canvas, para los lectores de

interesados en conocer de cerca esta nueva utilidad de HTML 5. Al menos esperamos dar a conocer las posibilidades del canvas y ofrecer algunos ejemplos que se puedan probar ya en los navegadores más modernos.

### 1.2.1.- Compatibilidad de canvas

El canvas se desarrolló inicialmente por Apple para su navegador Safari y luego fue utilizado y estandarizado por la organización WHATWG para incorporarlo a HTML 5. Posteriormente también ha sido adoptado por navegadores como Firefox y Opera.

Por lo que respecta a Chrome, es un navegador que utiliza el mismo motor de renderizado que Safari, por lo que también soporta el elemento Canvas.

De entre los navegadores más habituales sólo nos queda por soportar canvas el siempre polémico Internet Explorer. La última versión del navegador en el momento de escribir este artículo, Internet Explorer 8, no soporta canvas con funciones nativas, pero existen diversos proyectos y plugins que pueden ampliar las funcionalidades del navegador para dar soporte este nuevo elemento del HTML 5. Por ejemplo, existe el proyecto Explorer Canvas en el que se ha preparado un plugin para que Explorer soporte el dibujo 2d que permite canvas.

Sin embargo, aunque en diversos frentes se ha comenzado a utilizar Canvas, la falta de soporte de Explorer hace que todavía no sea muy recomendable su incorporación a las aplicaciones web, ya que la mayoría de los usuarios, que utilizan explorer, no podrían ver las partes de la página donde se utiliza canvas. Esta situación se espera que cambie durante los próximos meses o años, puesto que la incorporación de canvas al HTML 5 ya es una realidad e Internet Explorer más tarde o temprano tendrá que dar soporte esta utilidad en su navegador, si no quiere que se descarte su utilización por parte de los usuarios que deseen acceder a los servicios web más avanzados.

En la actualidad febrero 2015, i.e. 9 soporta canvas. Visitar la web: http://caniuse.com/#feat=canvas.

### 1.2.2.- Polémica por la propiedad intelectual de Canvas

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 HTML 5.

Afortunadamente 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 canvas dentro del nuevo estándar del lenguaje HTML.

### 1.2.3.- Aplicaciones de uso de Canvas

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 disponemos con el plugin de Flash, en lo que respecta a renderización de contenidos dinámicos. Las aplicaciones pueden ser grandes como podamos imaginar, desde juegos, efectos dinámicos en interfaces de usuario, editores de código, editores gráficos, aplicaciones, efectos 3D, etc.

Actualmente algunas de las aplicaciones más novedosas para para la web utilizan ya canvas para su funcionamiento, donde se puede destacar Bespin, un editor de código de Mozilla, o Google Wave. En la entrada de la Wiki sobre Canvas podemos encontrar diversos enlaces a sitios web con ejemplos de uso de este elemento del HTML 5.

En los siguientes artículos veremos ejemplos de uso de canvas y explicaremos cómo podemos utilizar nosotros mismos esta nueva herramienta del HTML 5.

# Parte 2:
# Primeros pasos con el elemento Canvas

Introducción al elemento Canvas y primeros ejemplos de dibujo. Aprendemos a crear Rectángulos y a configurar los colores que se utilizan en el dibujo.

# 2.1.- Ejemplo de dibujo con el API de canvas

*Un primer ejemplo de dibujo en un elemento canvas de HTML 5 con el API de canvas y Javascript.*

En el artículo anterior explicamos qué era el elemento canvas del HTML 5, así que podemos pasar a ver un primer ejemplo de utilización del mismo. Para comenzar realizaremos un ejemplo de dibujo de dos rectángulos con distintos colores, que realizaremos utilizando el un par de funciones del API de dibujo en canvas mediante Javascript. Claro que el elemento canvas tiene muchas cosas que debemos conocer para ir soltándonos en su manejo, pero al menos podremos ver una primera prueba para ir abriendo boca.

El ejemplo se basa en dos partes, primero una en la que colocaremos un lienzo canvas en un lugar de nuestra página, con la etiqueta HTML "CANVAS" y luego otra parte en la que dibujaremos dentro de ese elemento los rectángulos con programación Javascript. Sobra decir que harán falta unos conocimientos al menos básicos de Javascript para poder trabajar con el canvas.

## 2.1.1.- Colocar el elemento HTML canvas

Ahora comencemos situando dentro del cuerpo de la página la etiqueta CANVAS. Esta etiqueta, como decíamos forma parte del estándar del HTML 5.

```
<canvas id="micanvas" width="200" height="100">
Este texto se muestra para los navegadores no compatibles con canvas.
<br>
Por favor, utiliza Firefox, Chrome, Safari u Opera.
</canvas>
```

El elemento tiene apertura y cierre y entre medias podemos escribir un texto que será lo que vean los usuarios que entren con navegadores que no soporten la etiqueta CANVAS.

Para especificar las características de este canvas tenemos varios atributos:

### Atributo id:

Para asignarle un nombre único y luego referirnos a este canvas desde Javascript.

### Atributos width y height:

Para indicar la anchura y altura del área del canvas.

Otros atributos se pueden colocar de manera opcional, como por ejemplo style, para indicar atributos de hojas de estilo para definir el aspecto del lienzo.

## 2.1.2.- Pintar en un canvas mediante Javascript

Existen diversas funciones ya listas para dibujar formas y trazados en un canvas. Nosotros podemos combinarlas para hacer dibujos más complejos. Pero en este caso vamos a empezar dibujando un par de sencillas formas. No obstante, como veremos también, debemos hacer antes unas comprobaciones para saber si el navegador que está ejecutando esta página es compatible con canvas.

Inicialmente el canvas está en blanco y cuando queremos pintar sobre él tenemos que acceder al contexto de renderizado del canvas, sobre el que podremos invocar distintos métodos para acceder a las funciones de dibujo. El proceso simplificado sería el siguiente:

```
//Recibimos el elemento canvas
var canvas = document.getElementById('micanvas');
//Accedo al contexto de '2d' de este canvas, necesario para dibujar
var contexto = canvas.getContext('2d');
//Dibujo en el contexto del canvas
contexto.fillRect(50, 0, 10, 150);
```

- Primero con el método getElementById() obtengo el elemento de la página que se pasa como parámetro, que es el canvas.
- Luego accedo al contexto 2D del canvas, que es el que tiene varios métodos que sirven para dibujar en el lienzo.
- Por último puedo ejecutar tantos métodos como desee sobre el contexto del canvas para pintar elementos en el lienzo.

Como decía, estas sentencias Javascript no son compatibles con todos los navegadores, por lo que habrá que hacer unas comprobaciones básicas, para saber si ejecutar o no las distintas instrucciones de dibujo. Veamos este código, un poco más elaborado, que hace las comprobaciones necesarias para no hacer nada en el caso que el navegador no sea compatible con canvas.

```
//Recibimos el elemento canvas
var elemento = document.getElementById('micanvas');
//Comprobación sobre si encontramos un elemento
//y podemos extraer su contexto con getContext(), que indica compatibilidad con canvas
if (elemento && elemento.getContext) {
//Accedo al contexto de '2d' de este canvas, necesario para dibujar
var contexto = elemento.getContext('2d');
if (contexto) {
//Si tengo el contexto 2d es que todo ha ido bien y puedo empezar a dibujar
//Comienzo dibujando un rectángulo
contexto.fillRect(0, 0, 150, 100);
//cambio el color de estilo de dibujo a rojo
contexto.fillStyle = '#cc0000';
//dibujo otro rectángulo
contexto.fillRect(10, 10, 100, 70);
}
}
```

El código está comentado para que se pueda entender más fácilmente.

- Ahora sólo falta una última cosa, que es ejecutar estas acciones sólo cuando la página esté cargada por completo y lista para recibirlas. Esto lo conseguimos con la el evento onload del body de la página:

```
<body onload="funcionDeDibujo()">
```

Claro que tendremos que crear la funcionDeDibujo() con el código anterior para operar con el canvas.

- O bien podemos utilizar este otro recurso para asignar el evento directamente desde un script Javascript:

```
window.onload = function(){
//instrucciones de dibujo en canvas
}
```

El código completo de nuestro primer ejemplo de uso de canvas sería el siguiente:

```
<html>
<head>
<title>Probando canvas</title>
<script>
window.onload = function( ){
        //Recibimos el elemento canvas
        var elemento = document.getElementById('micanvas');
        //Comprobación sobre si encontramos un elemento
        //y podemos extraer su contexto con getContext(), que indica compatibilidad con canvas
        if (elemento && elemento.getContext) {
        //Accedo al contexto de '2d' de este canvas, necesario para dibujar
        var contexto = elemento.getContext('2d');
        if (contexto) {
        //Si tengo el contexto 2d es que todo ha ido bien y puedo empezar a dibujar en el canvas
        //Comienzo dibujando un rectángulo
        contexto.fillRect(0, 0, 150, 100);
        //cambio el color de estilo de dibujo a rojo
        contexto.fillStyle = '#cc0000';
        //dibujo otro rectángulo
        contexto.fillRect(10, 10, 100, 70);
        }
        }
        }
</script>
</head>
<body>
        <canvas id="micanvas" width="200" height="100">
        Este texto se muestra para los navegadores no compatibles con canvas.
        <br>
        Por favor, utiliza Firefox, Chrome, Safari u Opera.
        </canvas>
</body>
</html>
```

La visualización del resultado sería:.



# 2.2.- Entender el lienzo de canvas

*Veremos cómo es el lienzo de un canvas, formado por un eje de coordenadas que podemos utilizar para posicionar todos los dibujos que queramos colocar en el canvas.*

Seguimos dando nuestros primeros pasos en el elemento canvas del HTML 5. Recordemos que en el anterior artículo vimos un primer ejemplo de dibujo en un canvas, así que ya tenemos una idea de las partes que integran el proceso para hacer un dibujo en el canvas:

• Colocar la etiqueta CANVAS en el cuerpo de la página

• Dibujar en el canvas utilizando un script en Javascript

En este artículo vamos a explicar las características de nuestro lienzo y las coordenadas con las que podemos movernos por él y realizar dibujos.

## 2.2.1.- Eje de coordenadas del canvas

Para posicionar elementos en el canvas tenemos que tener en cuenta su eje de coordenadas en dos dimensiones, que comienza en la esquina superior izquierda del lienzo.

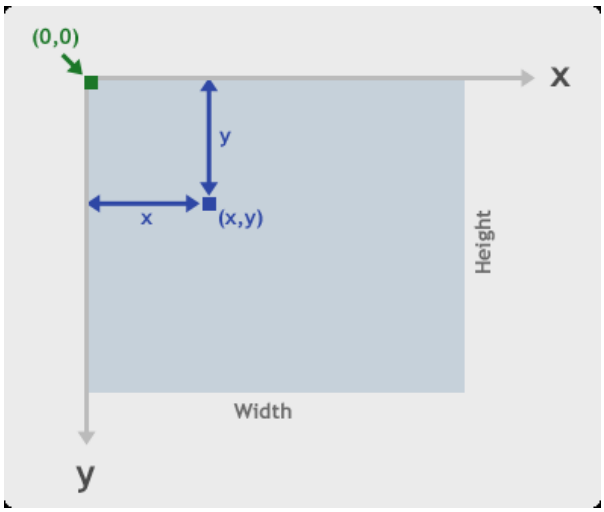
El lienzo producido por canvas tendrá unas dimensiones indicadas con los atributos width y height en la etiqueta CANVAS.

Por tanto, la esquina superior izquierda será el punto (0,0) y la esquina inferior derecha el punto definido por (width-1,height-1), o sea, el punto máximo de coordenadas marcado por su anchura y altura.

**Nota:** Hemos indicado que el punto de la esquina inferior derecha es (width-1,height-1) porque las coordenadas comienzan en (0,0), luego la coordenada final en anchura y altura será 1 menos el tamaño máximo de width y height definido en la etiqueta CANVAS. Por ejemplo, si la anchura es 50 y la altura es 100, entonces las coordenadas van desde (0,0) hasta (49,99).

Podemos ver el siguiente diagrama para tener una idea exacta de las dimensiones y coordenadas en un canvas.

Cualquier punto dentro del canvas se calcula con la coordenada (x,y), siendo que la x crece según los pixel a la derecha y la y con los pixel hacia abajo.



Para dibujar cualquier tipo de forma en el canvas necesitaremos posicionarla con respecto a las coordenadas que acabamos de ver. En el ejemplo del artículo anterior, vimos que para dibujar un rectángulo necesitamos varios valores:

```
contexto.fillRect(10, 10, 100, 70);
```

Los dos primeros parámetros eran las coordenadas x e y de la esquina superior izquierda del rectángulo. Los dos últimos parámetros son los valores de anchura y altura del mismo.

Pero hay otras formas que se pueden dibujar en un canvas que requieren el uso de métodos con un poco más elaborados que el dibujo de un rectángulo. Lo veremos más adelante.

## 2.2.2.- Segundo ejemplo de trabajo con canvas

Para seguir familiarizándonos con el elemento canvas y el procedimiento de dibujo mediante Javascript vamos a ver un segundo ejemplo, también sencillo.

Vimos que parte del código Javascript necesario era para realizar las comprobaciones pertinentes a fin de asegurarnos que el navegador es compatible con canvas cuando queremos extraer el contexto del canvas antes de empezar a dibujar. En este ejemplo realizaremos una función para resumir estas tareas que podremos utilizar a lo largo de este manual.

```
//Recibe un identificador del elemento canvas y carga el canvas
//Devuelve el contexto del canvas o FALSE si no ha podido conseguirse
function cargaContextoCanvas(idCanvas){
        var elemento = document.getElementById(idCanvas);
        if(elemento && elemento.getContext){
        var contexto = elemento.getContext('2d');
        if(contexto){
        return contexto;
        }
        }
        return FALSE;
}
```

Podremos invocar esta función y evaluar su resultado para saber si se obtuvo o no el contexto del canvas.

```
var contexto = cargaContextoCanvas('micanvas');
if(contexto){
//Tengo el contexto, entonces dibujo sobre el canvas
}
```

En este ejemplo vamos a dibujar un par de rectángulos, uno azul y otro amarillo. Ya vimos en el anterior ejemplo cómo se hacían formas rectangulares y también aprendimos a cambiar el color de la forma. Para mostrar otra de las posibilidades del canvas vamos a mostrar cómo hacer un color semitransparente.

```
//cambio el color de dibujo a azul
contexto.fillStyle = '#6666ff';
//dibujo un rectángulo azul
contexto.fillRect(10,10,50,50);
//cambio el color a amarillo con un poco de transparencia
contexto.fillStyle = 'rgba(255,255,0,0.7)';
//pinto un rectángulo amarillo semitransparente
contexto.fillRect(35,35,50,50);
```

Si nos fijamos, cuando se cambió el color a amarillo se especificó el color con RGBA, esto significa que estamos indicando también el canal Alpha, que indica el grado de transparencia desde 0 (totalmente transparente) a 1 (totalmente opaco).

Aclaramos de cualquier forma que todas estas funciones de dibujo las explicaremos con detenimiento más adelante.

El código completo de este segundo ejemplo se puede ver a continuación:

```
<html>
<head>
<title>Canvas segundo ejemplo</title>
<script>
//Recibe un identificador del elemento canvas y carga el canvas
//Devueve el contexto del canvas o FALSE si no ha podido conseguise
function cargaContextoCanvas(idCanvas){
var elemento = document.getElementById(idCanvas);
if(elemento && elemento.getContext){
var contexto = elemento.getContext('2d');
if(contexto){
return contexto;
}
}
return FALSE;
}
window.onload = function(){
//Recibimos el elemento canvas
var contexto = cargaContextoCanvas('micanvas');
if(contexto){
//Si tengo el contexto
//cambio el color de dibujo a azul
contexto.fillStyle = '#6666ff';
//dibujo un rectángulo azul
contexto.fillRect(10,10,50,50);
//cambio el color a amarillo con un poco de transparencia
contexto.fillStyle = 'rgba(255,255,0,0.7)';
//pinto un rectángulo amarillo semitransparente
contexto.fillRect(35,35,50,50);
}
}
</script>
</head>
<body>
<canvas id="micanvas" width="100" height="100">
Tu navegador no soporta canvas.
</canvas>
</body>
</html>
```

Visualizar ejemplo:

# 2.3.- Dibujar rectángulos en un Canvas

*Cómo dibujar cuadrados y rectángulos en un elemento Canvas de HTML mediante la función fillRect() y strokeRect() de Javascript, válida en HTML 5.*

Canvas es uno de los nuevos elementos disponibles en HTML 5, que sirve para dibujar cosas en un lienzo de la página. Como ya se explicó anteriormente, canvas es un elemento sobre el que dibujamos por medio de sentencias en el lenguaje de programación Javascript. Sin embargo, por el momento todos los navegadores no son compatibles con este nuevo componente, por lo que tendremos que hacer comprobaciones para no ejecutar en los navegadores ninguna instrucción que pueda dar errores por problemas de compatibilidad.

En este artículo vamos a explicar cómo podemos utilizar las funciones fillRect() y strokeRect() para dibujar rectángulos en la página y además vamos a implementar una pequeña interacción por medio de un enlace, que al pulsarlo ejecutará una función Javascript para borrar el contenido del canvas con la función clearRect().

## 2.3.1.- Función fillRect()

Esta función, perteneciente al objeto contexto de un elemento canvas, sirve para dibujar rectángulos rellenos de color.

Recibe cuatro parámetros, con este esquema:

```
fillRect(x,y,anchura,altura)
```

Esto dibuja un rectángulo cuya esquina superior izquierda está en el punto (x,y) y cuyas dimensiones son altura x anchura.

El color de relleno no lo especificamos en la propia función, sino que es el color que se tenga configurado en ese momento como color de relleno, que se indica con la propiedad fillStyle del contexto del canvas, asignando por ejemplo el RGB de un color.

**Nota:** Como vimos en el artículo Entender el lienzo de canvas, antes de ejecutar este método, necesitamos acceder al contexto de un canvas, para luego invocar al método sobre ese objeto. Por ejemplo, veamos el siguiente código:

```
for (i=0;i<=100;i+=10){
contexto.fillRect(i,i,5,5);
}
```

Esto dibujaría una serie de rectángulos, comenzando en la posición (0,0) y continuando con posiciones siempre de 10 píxeles de distancia en ambas coordenadas: (10,10), (20,20) ... Acabando en la coordenada (100,100). Todos los rectángulos serán de 5 píxeles de alto y ancho, luego realmente son cuadrados.

## 2.3.2.- Función strokeRect()

Esta función sirve para dibujar simplemente la silueta de un rectángulo, es decir, sólo su borde. El esquema de parámetros es el siguiente:

```
strokeRect(x,y,anchura,altura)
```

Dibuja el borde de un rectángulo comenzando en la posición (x,y) para su esquina superior izquierda y con las dimensiones de altura x anchura.

```
for (i=100;i>=0;i-=10){
contexto.strokeRect(i,100-i,5,5);
}
```

Con el código anterior también dibujamos una serie de cuadrados en el canvas, aunque en esta ocasión sólo la silueta sin estar rellenos de color, de 5 píxeles de anchura y altura y con distintas coordenadas que producimos al hacer el bucle for.

De manera similar, para definir el color del borde del rectángulo, utilizamos la propiedad strokeStyle del objeto del contexto del canvas, a la que podemos asignar el valor RGB que deseemos para el borde de los cuadrados o aquello que vayamos a dibujar en el canvas.

## 2.3.3.- Función clearRect()

Esta función nos sirve para borrar áreas rectangulares de un canvas y hacerlas totalmente transparentes o sin contenido gráfico. Funciona de manera similar a los rectángulos:

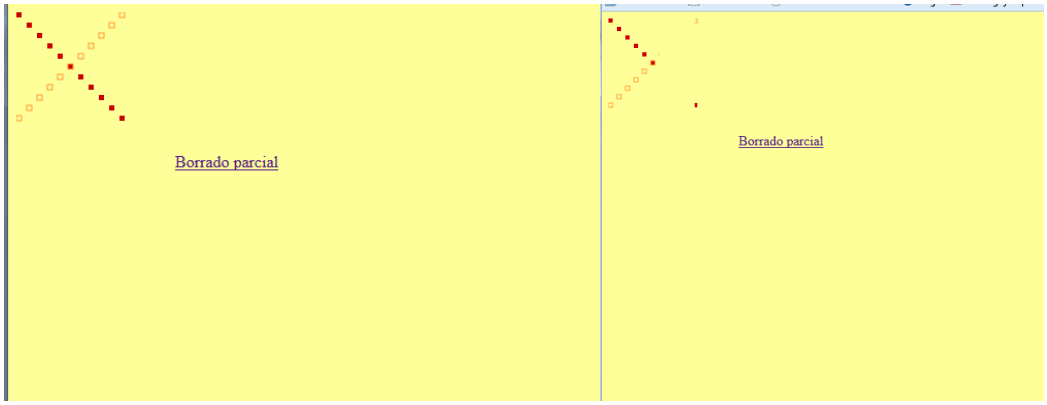
```
clearRect(x,y,anchura,altura)
```

El color aquí no importa mucho, porque es simplemente el color del fondo del contenedor HTML donde hayamos colocado el canvas.

## 2.3.4.- Ejemplo de creación de rectángulos en canvas.

## Utilizando todo lo expuesto anteriormente, crea un html5, en el que se dibujen rectángulos rellenos, rectángulos huecos, estén dispuestos en diagonal y sobre el texto “Borrado parcial” se ejecute la función borrar.

- Crear la function cargaContextoCanvas(idCanvas)
- Llamar a dicha función.
- Utilizar fillStyle para dar color a los rectángulos rellenos.
- Utilizar fillRect(x,y,width,height)
- Utilizar strokeStyle para dar color a los rectángulos huecos.
- Utilizar strokeRect(x,y,width,height)
- Crear una function borrar_parcial, que utilice clearRect(x,y,width,height) para borrar solo la parte de los rectángulos huecos.




# 2.4.- Trabajando con color de relleno y de trazado en canvas

*Atributos fillStyle para el color de relleno y strokeStyle para el color de trazado.*

Estamos aprendiendo acerca del elemento Canvas del HTML 5 y vamos a presentar un artículo que nos servirá para darle un toque personal a nuestros diseños utilizando un poco de color.

Para trabajar con color en los elementos canvas tenemos varias posibilidades, pero de momento vamos a aprender a modificar el color con el que se rellena o se dibuja trazados. Como vimos anteriormente, al dibujar rectángulos, podemos elegir entre dibujar sólo su contorno o dibujarlos rellenos de color (y luego veremos que esto es así con otros tipos de caminos). Ahora veremos que existen dos atributos del contexto del canvas que sirven para definir el color de relleno y el color de trazado a la hora de dibujar rectángulos u otros caminos.

## 2.4.1.- Cambiar el color de relleno con fillStyle

Existe un atributo del contexto del canvas que almacena el color que se utilizará al rellenar elementos. Cambiar el color de relleno es tan sencillo como asignarle valores distintos, de colores en RGB, con lo que conseguiremos que la próxima vez que se rellene de color se haga con ese nuevo valor asignado.

```
ctx.fillStyle = '#990000';
```

Suponiendo que tenemos el objeto contexto de un canvas en la variable ctx, con la anterior línea estamos solicitando al elemento canvas que la próxima vez que se rellene el color se haga en rojo oscuro.

## 2.4.2.- Cambiar el color de trazado con strokeStyle

Ahora bien, cuando dibujamos podemos elegir hacer sólo un trazado del rectángulo, u otro tipo de camino y para ello se utilizará otro color que podemos definir con strokeStyle. El atributo strokeStyle funciona de la misma manera que fillStyle, pero con la salvedad que servirá para indicar el color del trazado.

```
ctx.strokeStyle = '#000099';
```

Con esa línea estamos marcando que el color de trazado sea azul oscuro. Por lo cual, la próxima vez que se haga un trazado la línea será de ese color.

## 2.4.3.- Otras notaciones para definir colores en canvas

Ahora podemos aprender cómo especificar colores en los elementos canvas. La verdad es que los que conozcan CSS no tendrán problema alguno para ello, ya que los formatos son exactamente los mismos. Por ejemplo, podremos utilizar estas notaciones.

- Color con nombre: "blue"
- Color con RGB hexadecimal, como se define en HTML: "#ff6600"
- Color con RGB en base decimal: "rgb(100, 25, 206)"
- Color RGBA (canal alpha o transparencia, como en CSS3): "rgba(255, 125, 0, 0.5)"
- Con RGB y porcentaje: "rgb(100%, 20%, 0)"