

**8. HTML5 AVANZADO**

Las tecnologías HMTL5 junto con Javascript

marcan uno de los caminos de evolución de las

aplicaciones web 2.0, es decir, influyen de forma

muy determinante sobre el futuro del diseño y

desarrollo web. Un paso muy importante que ha

permitido esta influencia es la implementación de

APIs (funciones de programación específicas que

realizan una determina tarea) por parte de los

navegadores modernos lo que ha permitido a los

desarrolladores web ahorrar tiempo y simplificar

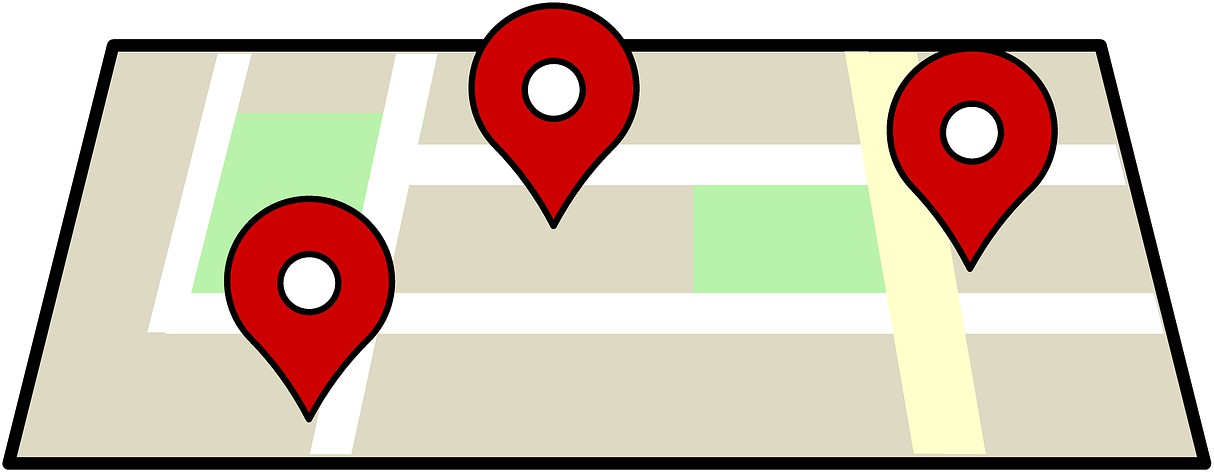
sus aplicaciones ya que sólo se tienen que preocupar en utilizar estos códigos en vez de

tenerlos que programar.

Existen infinidad de APIs y de aplicaciones que usan HTML5 y Javascript por lo que en

este tema vamos a ver solamente tres de ellas que nos van a ayudar a hacernos una idea de

cómo funcionan y cómo se pueden utilizar.



**8.1. INTRODUCCIÓN**

De las nuevas funcionalidades avanzadas importantes que ofrece HTML5 podemos

destacar el uso de APIs ya implementadas en los navegadores a través de scripts

javascript normalmente. Esto permite aumentar las prestaciones de las páginas web sin

tener que recurrir a plugins.

Sin entrar en muchos detalles técnicos se

define como API, interfaz de programación

de aplicaciones, a las funciones o métodos

(programación orientada a objetos) que

utiliza otro software (programa, página web,

etc.) para realizar una determinada tarea.

Esto evita que el programador de ese

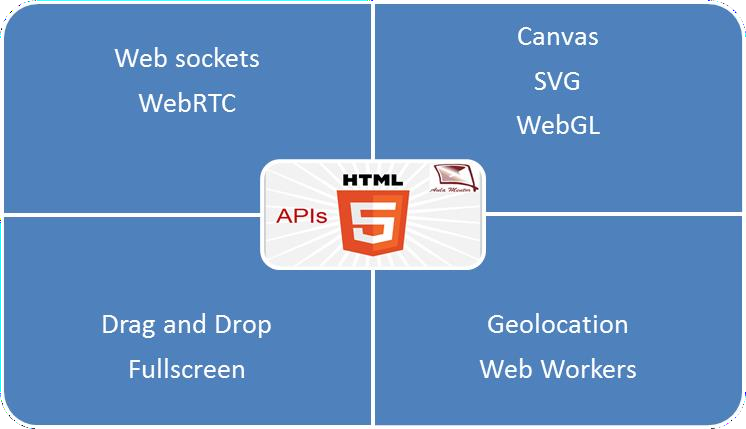
software tenga que implementar o

programar esas funciones que ofrecen las

APIs y solamente las utilice sin conocer sus

detalles internos.

Ejemplos de APIs



 **Geolocation**

Se emplea para conocer la posición geográfica del usuario.

 **Drag and Drop**

Se emplea para realizar la operación de coger un objeto, trasladarlo y dejarlo en

otro lugar.

 **Local Storage**

Se emplea para crear un espacio de almacenamiento local en el cual se guardan

datos. Es la opción que ofrece HTML5 para suplir las “cookies”.

 **Application Cache**

Se emplea para guardar una copia de la aplicación web en una memoria intermedia

o caché y poder emplearla sin tener conexión a Internet.

 **Web Workers**

Se emplea para ejecutar scripts en una página HTML5 en segundo plano o

background.

 **Server-Sent Events**

Se emplea para actualizar páginas web desde un servidor.

 **Canvas**

Se emplea para dibujar gráficos en tiempo real a través de scripts javascript.

**8.2. GEOLOCATION**

Esta API se emplea para determinar la posición geográfica de un usuario devolviendo su longitud y latitud.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **API** |  |  |  |  |  |
| **geolocation** | 9.0 | 5.0 | 3.5 | 5.0 | 16 |

El método **getCurrentPosition()** de **geolocation** permite saber la posición del usuario.

!DOCTYPE html>

<html lang="es-es">

<head>

<title>Gelocation</title>

<meta charset="UTF-8">

<style>

body { background-color: #ccffcc; }

</style>

</head>

<body>

<p>Pulsa el botón para saber tus coordenadas</p>

<button onclick="coordenadas()">Pulsa</button>

<p id="párrafo"></p>

<script>

var elemento = document.getElementById("párrafo");

function coordenadas() {

if (navigator.geolocation) { navigator.geolocation.getCurrentPosition(posición);

} else {

elemento.innerHTML = "Geolocation no es soportada por tu navegador";

}

}

function posición(lugar) {

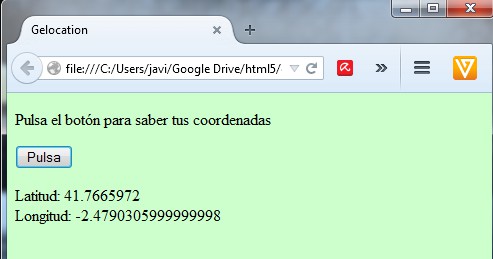
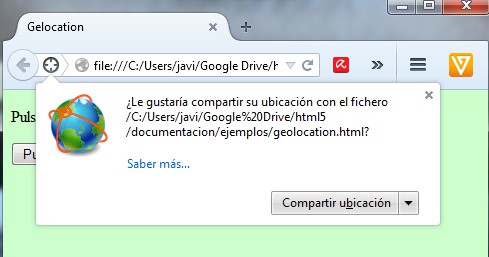
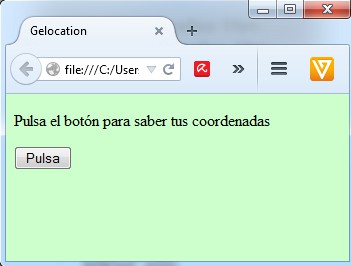
elemento.innerHTML = "Latitud: " + lugar.coords.latitude + "<br>Longitud: " + lugar.coords.longitude;

}

</script>

</body>

</html>



Ahora podemos modificar la página anterior y usar google maps para visualizar nuestra posición en un mapa

<!DOCTYPE html>

<html lang="es-es">

<head>

<title>Gelocation con google</title>

<meta charset="UTF-8">

<style>

body { background-color: #ccffcc; }

</style>

</head>

<body>

<p>Pulsa el botón para saber tus coordenadas</p>

<button onclick="coordenadas()">Pulsa</button>

<section id="contenedor"></section>

<p id=”párrafo”></p>

<script>

var elemento = document.getElementById("párrafo");

function coordenadas() {

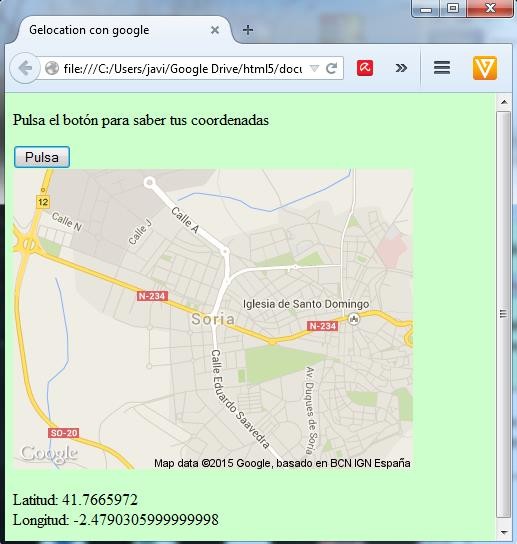
if (navigator.geolocation) {

navigator.geolocation.getCurrentPosition(posición);

} else {

elemento.innerHTML = "Geolocation no es soportada por tu navegador";

}



}

function posición(lugar) {

elemento.innerHTML = "Latitud: " + lugar.coords.latitude + "<br>Longitud: " + lugar.coords.longitude;

var mapa=lugar.coords.latitude + "," + lugar.coords.longitude; var

imagenmap[a="http:](http://maps.googleapis.com/maps/api/staticmap?center)//m[aps.googleapis.com/maps/api/staticmap?center="](http://maps.googleapis.com/maps/api/staticmap?center)

+mapa+"&zoom=14&size=400x300&sensor=false"; document.getElementById("contenedor").innerHTML = "<img

src='"+imagenmapa+"'>";

}

</script>

</body>

</html>

# DRAG AND DROP

Esta API permite “arrastar y soltar” (**drag and drop**) un objeto para llevarlo de un lugar a otro dentro de la misma ventana o entre ventanas. Normalmente esta operación consta de tres partes:

* Se posiciona el ratón encima del objeto y se pulsa, normalmente, el botón izquierdo por lo que el objeto queda “agarrado”.
* Se arrastra desde la posición inicial en la que estaba dicho objeto hasta el lugar de destino.
* Se suelta el objeto en el lugar de destino.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **API** |  |  |  |  |  |
| **Drag and Drop** | 9.0 | 5.0 | 3.5 | 6.0 | 12.0 |



Para que a un elemento HTML5 se le pueda aplicar esta API, es decir, hacerlo “arrastrable” hay que agregar, en primer lugar, el atributo **draggable** con el valor **true**.

*El API* ***Drag and Drop*** *forma parte del standard HTML5*

<img **draggable=”true”**>

Ejemplo:

<!DOCTYPE HTML>

<html>

<head>

<style>

body { background-color: #ccffcc; } section {

width:700px; height:300px;

border: 2px solid #ccddee; background-image: url("tablero.png");

background-repeat: no-repeat; background-size:700px 600px; background-position: center;

}

img { display:inline;

width: 88px; height: 73px;

}

</style>

<script>

function permitir(evento) { evento.preventDefault();

}

function arrastrar(evento) { evento.dataTransfer.setData("text", evento.target.id);

}

function soltar(evento) { evento.preventDefault();

var data = evento.dataTransfer.getData("text"); evento.target.appendChild(document.getElementById(data));

}

</script>

</head>

<body>



<img id="caballo" src="caballo.jpg" draggable="true" ondragstart="arrastrar(event)">

</body>

</html>

<section ondrop="soltar(event)" ondragover="permitir(event)"></section>

<br>

<img id="torre" src="torre.jpg" draggable="true" ondragstart="arrastrar(event)">

<h1> Deja la torre y el caballo en el tablero</h1>



En este ejemplo partimos de dos imágenes (una torre y un caballo del juego del ajedrez) que están debajo de un tablero de ajedrez. Utilizando dos operaciones “drag and drop”

colocamos primero el caballo y luego la torre dentro del tablero que es la imagen de fondo de un elemento <section>.

Para entender mejor el ejemplo y comprender el fundamento del evento “arrastrar y soltar” vamos a explicarlo por partes o pasos:

* *Hacer las imágenes de la torre y del caballo “arrastrables”*: Se agrega a los elementos <img> el atributo **draggable** con el valor “true”

|  |
| --- |
| <img id="torre" src="torre.jpg" **draggable="true"**  ondragstart="arrastrar(event)">  <img id="caballo" src="caballo.jpg" **draggable="true**" ondragstart="arrastrar(event)"> |

* *Se especifica lo que va a suceder cuando se inicia la operación de arrastre de cualquiera de las dos imágenes:* El atributo **ondragstart** llama a la función **arrastrar**() que especifica qué elemento se puede arrastrar y el método **dataTransfer.setData()** define el tipo y la identificación del elemento arrastrado. En este caso, el tipo de datos es “texto” (text) y el valor del id es “torre” y “caballo”.

|  |
| --- |
| function arrastrar(evento) { evento.**dataTransfer.setData("text", evento.target.id);**  }  <img id="torre" src="torre.jpg" draggable="true"  **ondragstart="arrastrar(event)**">  <img id="caballo" src="caballo.jpg" draggable="true"  **ondragstart="arrastrar(event)"**> |

* *Dónde dejar el elemento que es arrastrado*: El evento **ondragover** específica donde puede ser dejado el elemento que es arrastrado.

Por defecto, no se puede dejar un elemento arrastrado dentro de otro elemento, por lo que, si queremos hacer esto, debemos emplear el método **preventDefault()** del evento **ondragover**.

|  |
| --- |
| function permitir(evento) {  **evento.preventDefault();**  }  <section ondrop="soltar(event)" **ondragover="permitir(event)**"></section> |

* *Soltar el elemento:* Cuando el elemento arrastrado es soltado se produce un evento “soltar” o “drop”. En el ejemplo se emplea la función “soltar”.

|  |
| --- |
| function **soltar(evento)** { evento.preventDefault();  var data = evento.dataTransfer.getData("text"); evento.target.appendChild(document.getElementById(data));  }  <section **ondrop="soltar(event)"** ondragover="permitir(event)"></section> |

# CANVAS

El nuevo elemento <canvas> de HTML5 permite dibujar gráficos de forma dinámica (que cambian en tiempo real) dentro de una página web a través de scripts escritos normalmente en javascript.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Etiqueta | <canvas> | | | | | | |
| Descripción | Elemento para dibujar gráficos | | | | | | |
| Elemento | Bloque | | | | | | |
| Navegadores que la  soportan |  |  |  |  |  |  |  |
| Chrome | Firefox | IExplorer | Ópera | Safari |  |
| 4.0 | 2.0 | 9.0 | 9.0 | 3.1 |  |
| Atributos | Globales y eventos | | | | | | |
| Atributos propios | height, width | | | | | | |
| Diferencia  entre html  4.01 y html5 | El elemento <canvas> es nuevo en HTML5 | | | | | | |



#### Ejemplo simple:

*Aunque* ***<canvas>*** *es un elemento también se emplea un API canvas para las operaciones de dibujo de gráficos*

<!DOCTYPE html>

<html lang="es-es"

<head>

<meta charset="UTF-8">

<title>canvas</title>

<style>

body { background-color: #ccffcc; } #dibujo {

border:3px solid brown;

}

</style>

</head>

<body>

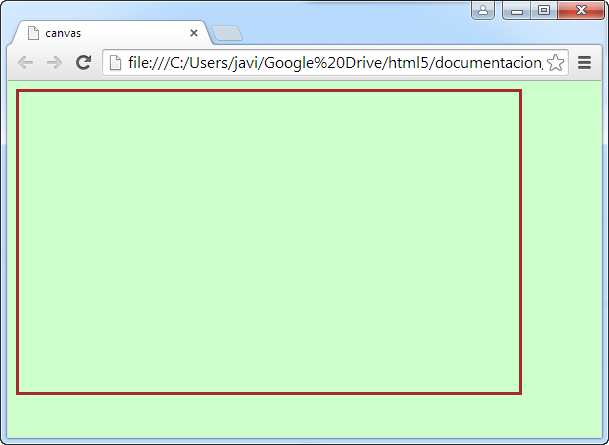
<canvas id="dibujo" height="300" width="500">

Tu navegador no soporta &lt;canvas&gt;

</canvas>

</body>

</html>



En este ejemplo simplemente hemos definido el elemento <canvas> con id igual a “dibujo” con una anchura de 500 pixeles, una altura de 300 pixeles y un borde de 3 pixeles de color marrón. Dentro de este elemento es donde vamos a “dibujar” a través de scripts javascript en los cuales se usan métodos de canvas.

#### Ejemplo de dibujo de un rectángulo:

*Todos los dibujos canvas deben ser dibujados dentro de scripts javascript normalmente*



<!DOCTYPE html>

<html lang="es-es"

<head>

<meta charset="UTF-8">

<title>canvas</title>

<style>

body { background-color: #ccffcc; } #dibujo {

border:3px solid brown;

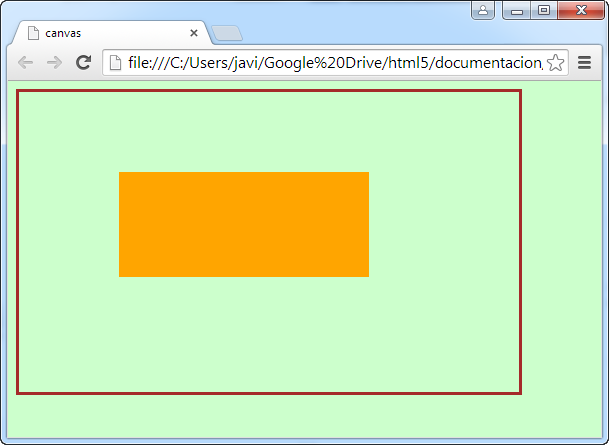
}

</style>

</head>

<body>

<canvas id="dibujo" height="300" width="500">



Tu navegador no soporta &lt;canvas&gt;

</canvas>

<script>

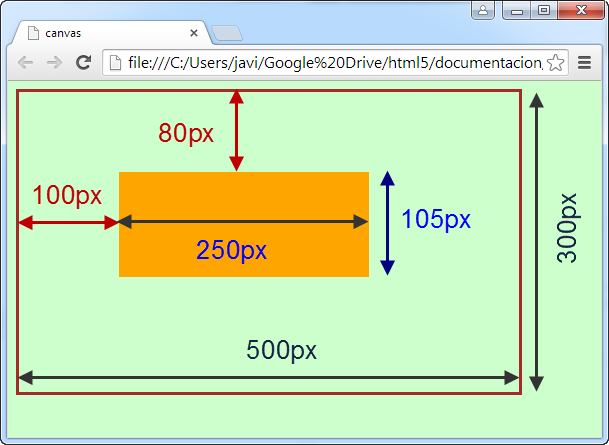
var gráfico = document.getElementById("dibujo");

var contexto = gráfico.getContext("2d"); contexto.fillStyle = "orange"; contexto.fillRect(100,80,250,105);

</script>

</body>

</html>



En este ejemplo creamos el elemento <canvas> con id igual a “dibujo” de 300 pixeles de alto por 500 de ancho que va a ser el área donde vamos a dibujar. A través de un script javascript se dibuja en el área anterior un rectángulo con relleno de color naranja (contexto.fillSyle=”orange”) que comienza en el punto 100 (100 pixeles desde el borde izquierdo del elemento “dibujo”), 80 (80 pixeles desde el borde superior del elemento “dibujo”) con una anchura de 250 pixeles y una altura de 105 pixeles (contexto.fillRect(100,80,250,105)) siguiendo estos pasos:

* *Buscar el elemento <canvas> con id igual a “dibujo*”: Esto se hace usando el método HTML DOM getElementById() y asignándolo a la variable “gráfico”

var gráfico = document.getElementById("dibujo");

* *Crear un objeto dibujable*: A través del método getContext() definimos el objeto “contexto” al que se puede aplicar propiedades y métodos de dibujo.

var contexto = gráfico.getContext(“2d”);

* *Dibujar dentro del elemento <canvas> con id igual a “dibujo”*: Siguiendo el ejemplo primero le asignamos el color de relleno naranja al objeto “contexto” a través de la propiedad fillStyle.

contexto.fillStyle = "orange";

Y, por último, dibujamos un rectángulo relleno con el color asignado a la propiedad fillStyle usando el método fillRect().

contexto.fillRect(100,80,250,105);

#### Coordenadas canvas

Un elemento <canvas> es un cuadrilátero con dos dimensiones (ancho y alto). Lo que dibujamos dentro del elemento <canvas> tiene unas coordenadas que lo posicionan en un lugar concreto. La esquina superior izquierda del elemento

<canvas> es el punto (0,0) donde la componente izquierda corresponde con la distancia del borde izquierdo del elemento <canvas> y la componente derecha es la distancia desde el borde superior. Si nos fijamos en la imagen superior vemos que el rectángulo tiene las coordenadas (100,80), es decir, esas coordenadas se corresponden con el punto desde el cual se empieza a dibujar la figura. El número de la izquierda (100) es la distancia al borde izquierdo y el de la derecha (80) es la distancia desde el borde superior.

#### Dibujando una línea recta

Para dibujar una línea recta dentro de un elemento <canvas> se utlilizan los métodos:

* + - * moveTo(izda, arriba): Define el punto inicial de la recta.
      * lineTo(izda,arriba): Define el punto final de la recta.
      * stroke(): Dibuja la línea Y la propiedad:
      * strokeStyle: Define el color de la línea.

#### Ejemplo:

|  |
| --- |
| <!DOCTYPE html>  <html lang="es-es"  <head>  <meta charset="UTF-8">  <title>Línea canvas</title>  <style>  body { background-color: #ccffcc; } #dibujo {  border:3px solid brown;  }  </style>  </head>  <body>  <canvas id="dibujo" height="300" width="500">  Tu navegador no soporta &lt;canvas&gt;  </canvas>  <script>  var gráfico = document.getElementById("dibujo"); var contexto = gráfico.getContext("2d"); contexto.moveTo(80,100); contexto.lineTo(300,200); contexto.strokeStyle="blue";  contexto.stroke(); |

|  |
| --- |
| </script>  </body>  </html> |
|  |
|  |

Como vemos en este ejemplo se define el elemento <canvas> y, dentro ya del

<script>, se usan las dos mismas declaraciones utilizadas para dibujar el

rectángulo anterior ya que hay que identificar el elemento <canvas> y hacerlo dibujable. Estas declaraciones son:

var gráfico = document.getElementById("dibujo"); var contexto = gráfico.getContext("2d");

Las siguientes declaraciones del script definen el punto inicial de la línea (contexto.moveTo(80,100);), el punto final de la misma (contexto.lineTo(300,200);), su color (contexto.strokeStyle="blue";) y se dibuja la recta (contexto.stroke();).

#### Ejemplo de dibujo de varias líneas en un elemento <canvas>:

|  |
| --- |
| <!DOCTYPE html>  <html lang="es-es"  <head>  <meta charset="UTF-8">  <title>Varias línea canvas</title>  <style>  body { background-color: #ccffcc; } #dibujo {  border:3px solid brown;  }  </style>  </head>  <body>  <canvas id="dibujo" height="300" width="500">  Tu navegador no soporta &lt;canvas&gt;  </canvas>  <script>  var gráfico = document.getElementById("dibujo"); var contexto = gráfico.getContext("2d");  // Líneas verticales  for (var x = 20; x < 500; x += 20) { contexto.moveTo(x, 0);  contexto.lineTo(x, 300);  }  //líneas horizontales  for (var y = 20; y < 300; y += 20) { contexto.moveTo(0, y); contexto.lineTo(500, y);  }  //dibujo de todas las líneas en color negro  contexto.strokeStyle = "black"; contexto.stroke();  </script> |

|  |
| --- |
| </body>  </html> |
|  |

En este ejemplo dibujamos un “grid” de líneas rectas tanto verticales como horizontales a través de dos bucles que separa las líneas rectas 20 pixeles entre sí. Como se ve sólo hace falta ejecutar una vez la propiedad strokeStyle y el método stroke() para que dibuje todas las líneas. Esto es debido a que, en realidad, “contexto” es un objeto y actúa como tal. Recordad también que las líneas que empiezan por “\\” son comentarios en javascript.

#### Dibujando un círculo

Para dibujar un círculo en canvas, además del método stroke() y de la propiedad strokeStyle vistos en el punto anterior, se usan los siguientes métodos:

* + - * beginPath(): Indica que se va a dibujar una figura que puede cerrarse o no como, por ejemplo, polígonos o círculos o parte de ellos.
      * arc(izda,dcha,radio,ángulo inicial, ángulo final, sentido): Sirve para definir un arco con centro en las coordenadas (izda, dcha) con un radio “radio”, empezando en el “ángulo inicial”, acabando en el “ángulo final” y si el sentido del dibujo es igual al de las agujas del reloj (true) o al revés (false).

Y la propiedad:

* + - * lineWidth: Se utiliza para indicar el grosor de la línea.

|  |
| --- |
| <!DOCTYPE html>  <html lang="es-es"  <head>  <meta charset="UTF-8">  <title>Círculo canvas</title> |

|  |
| --- |
| <style>  body { background-color: #ccffcc; } #dibujo {  border:3px solid brown;  }  </style>  </head>  <body>  <canvas id="dibujo" height="300" width="500">  Tu navegador no soporta &lt;canvas&gt;  </canvas>  <script>  var gráfico = document.getElementById("dibujo"); var círculo = gráfico.getContext("2d");  círculo.beginPath(); círculo.arc(140,90,50,0,2\*Math.PI); círculo.strokeStyle = "blue"; círculo.stroke();  </script>  </body>  </html> |
|  |

|  |
| --- |
|  |

#### Ejemplo que muestra varios dibujos con el método *arc*

|  |
| --- |
| <!DOCTYPE html>  <html lang="es-es"  <head>  <meta charset="UTF-8">  <title>Círculo canvas</title>  <style>  body { background-color: #ccffcc; } #dibujo {  border:3px solid brown;  }  </style>  </head>  <body>  <canvas id="dibujo" height="300" width="500">  Tu navegador no soporta &lt;canvas&gt;  </canvas>  <script>  var gráfico = document.getElementById("dibujo"); var círculo = gráfico.getContext("2d");  var círculo1 = gráfico.getContext("2d"); var círculo2 = gráfico.getContext("2d"); var círculo3 = gráfico.getContext("2d"); var círculo4 = gráfico.getContext("2d");  círculo.beginPath();  círculo.arc(140,90,50,0,2\*Math.PI); círculo.strokeStyle = "blue"; |

|  |
| --- |
| círculo.stroke();  círculo1.beginPath(); círculo1.arc(350,90,50,0,Math.PI); círculo1.strokeStyle = "blue"; círculo1.stroke();  círculo2.beginPath(); círculo2.arc(140,225,50,Math.PI,2\*Math.PI); círculo2.strokeStyle = "blue"; círculo2.stroke();  círculo3.beginPath(); círculo3.arc(350,225,50,0,Math.PI,true); círculo3.strokeStyle = "blue"; círculo3.stroke();  </script>  </body>  </html> |
|  |

|  |
| --- |
|  |

En estos ejemplos se emplea la constante “Math.PI” para la asignación de ángulos que se corresponde con el número pi (3,1415….) representado en radianes perteneciente al objeto predefinido “Math” de javascript. Este objeto permite operaciones matemáticas como redondeos, números aleatorios, etc.

#### Gradientes o difuminados

En canvas se pueden rellenar textos o figuras como círculos, rectángulos, etc., de una determinada zona del elemento <canvas> con tonos difuminados o gradientes a través de los siguientes métodos:

* + - * createLinearGradient(izda,arriba,izda1, arriba1): Habilita una zona del elemento <canvas> donde se puede aplicar un difuminado lineal que empieza en el punto (izda, arriba) y acaba en el punto (izda1,arriba1).
      * createLinearGradient(izda,arriba,radio,izda1,arriba1,radio1): Habilita una zona del elemento <canvas> donde se puede aplicar un difuminado concéntrico que empieza en el punto (izda,arriba) con radio “radio” y acaba en el punto (izda1, arriba1) con radio “radio1”.
      * addColorStop(número de orden, color): Especifica el color que se difumina y el orden dentro del relleno. El “número de orden” estará comprendido entre 0 y 1 pudiéndose emplear números decimales como, por ejemplo, 0.5.

Para emplear los gradientes en una figura se asigna el objeto que se ha creado como gradiente a las propiedades fillStyle o strokeStyle.

#### Ejemplo de difuminado lineal de izquierda a derecha de una zona donde se dibuja un rectángulo:

|  |
| --- |
| <!DOCTYPE html>  <html lang="es-es"  <head>  <meta charset="UTF-8">  <title>Rectángulo difuminado canvas</title>  <style>  body { background-color: #ccffcc; } #dibujo {  border:3px solid brown;  }  </style>  </head>  <body>  <canvas id="dibujo" height="300" width="500">  Tu navegador no soporta &lt;canvas&gt;  </canvas>  <script>  var gráfico = document.getElementById("dibujo"); var rectángulo = gráfico.getContext("2d");  //creando difuminado  var difuminado=rectángulo.createLinearGradient(0,0,300,0); difuminado.addColorStop (0,”blue”);  difuminado.addColorStop (1,”yellow”);  //usando difuminado  rectángulo.fillStyle= difuminado; rectángulo.fillRect (50,100,200,150);  </script>  </body>  </html> |

|  |
| --- |
|  |

En este ejemplo se define un difuminado de izquierda a derecha en la zona que va desde el borde izquierdo hasta 300 puntos de ancho que cambia de color azul a amarillo y dentro de esa zona se dibuja un rectángulo al cual se aplica el difuminado.

#### Ejemplo de difuminado lineal de arriba a abajo de una zona donde se dibuja un rectángulo:

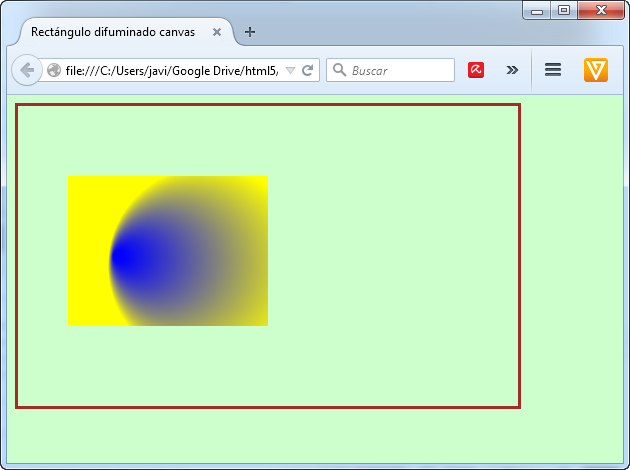
|  |
| --- |
| <!DOCTYPE html>  <html lang="es-es"  <head>  <meta charset="UTF-8">  <title>Rectángulo difuminado canvas</title>  <style>  body { background-color: #ccffcc; } #dibujo {  border:3px solid brown;  }  </style>  </head>  <body>  <canvas id="dibujo" height="300" width="500"> |

|  |
| --- |
| Tu navegador no soporta &lt;canvas&gt;  </canvas>  <script>  var gráfico = document.getElementById("dibujo"); var rectángulo = gráfico.getContext("2d");  //creando difuminado  var difuminado=rectángulo.createLinearGradient(0,0,0,300); difuminado.addColorStop (0,"blue"); difuminado.addColorStop (1,"yellow");  //usando difuminado  rectángulo.fillStyle= difuminado; rectángulo.fillRect (50,70,200,150);  </script>  </body>  </html> |
|  |

En este ejemplo se define un difuminado de arriba a abajo en la zona que va desde el borde superior hasta 300 puntos más abajo que cambia de color azul a amarillo y dentro de esa zona se dibuja un rectángulo al cual se aplica el difuminado.

#### Ejemplo de difuminado radial de una zona donde se va a dibujar un rectángulo:

|  |
| --- |
| <!DOCTYPE html>  <html lang="es-es"  <head>  <meta charset="UTF-8">  <title>Rectángulo difuminado canvas</title>  <style>  body { background-color: #ccffcc; } #dibujo {  border:3px solid brown;  }  </style>  </head>  <body>  <canvas id="dibujo" height="300" width="500">  Tu navegador no soporta &lt;canvas&gt;  </canvas>  <script>  var gráfico = document.getElementById("dibujo"); var rectángulo = gráfico.getContext("2d");  //creando difuminado var  difuminado=rectángulo.createRadialGradient(100,150,5,190,160,100); difuminado.addColorStop (0,"blue"); difuminado.addColorStop (1,"yellow");  //usando difuminado  rectángulo.fillStyle= difuminado; rectángulo.fillRect (50,70,200,150);  </script>  </body>  </html> |



|  |
| --- |
|  |

#### Texto

Los métodos y propiedades más importantes que se emplean para dibujar textos en canvas son:

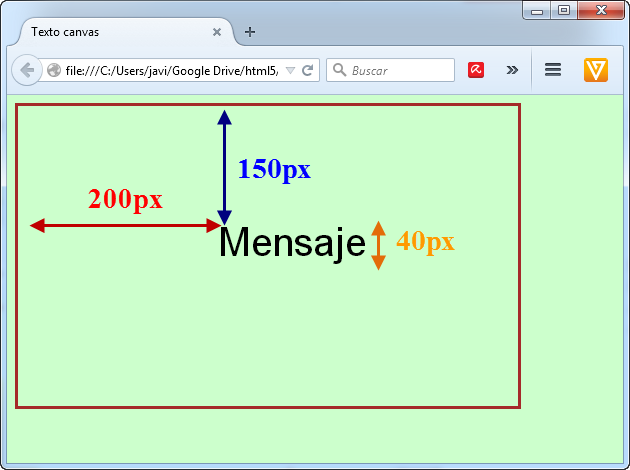
* + - * font: Define las propiedades de la fuente del texto tales como tamaño, familia, etc.
      * fillText(texto, izda, arriba): Dibuja texto con relleno donde el par de números izda y arriba representan las coordenadas de donde se va a empezar a escribir.
      * strokeText(texto,izda,arriba): Dibuja texto sin relleno. Al igual que fillText() izda y arriba son las coordenadas de donde empieza el texto.
      * textAlign: Alinea el texto.

**Ejemplo de texto con relleno:**

|  |
| --- |
| <!DOCTYPE html>  <html lang="es-es"  <head>  <meta charset="UTF-8">  <title>Texto canvas</title>  <style>  body { background-color: #ccffcc; } #dibujo {  border:3px solid brown;  } |

|  |
| --- |
| </style>  </head>  <body>  <canvas id="dibujo" height="300" width="500">  Tu navegador no soporta &lt;canvas&gt;  </canvas>  <script>  var gráfico = document.getElementById("dibujo"); var texto = gráfico.getContext("2d");  texto.font = "40px Arial"; texto.fillText("Mensaje", 200,150);  </script>  </body>  </html> |
|  |

|  |
| --- |
|  |

**Ejemplo de texto con letra hueca:**

|  |
| --- |
| <!DOCTYPE html>  <html lang="es-es"  <head>  <meta charset="UTF-8">  <title>Texto canvas</title>  <style>  body { background-color: #ccffcc; } #dibujo {  border:3px solid brown;  }  </style>  </head>  <body>  <canvas id="dibujo" height="300" width="500">  Tu navegador no soporta &lt;canvas&gt;  </canvas>  <script>  var gráfico = document.getElementById("dibujo"); |

|  |
| --- |
| var texto = gráfico.getContext("2d"); texto.font = "40px Arial"; texto.strokeText("Mensaje", 200,150);  </script>  </body>  </html> |
|  |

**Ejemplo de texto centrado y de color azul:**

|  |
| --- |
| <!DOCTYPE html>  <html lang="es-es"  <head>  <meta charset="UTF-8">  <title>Texto canvas</title>  <style>  body { background-color: #ccffcc; } #dibujo {  border:3px solid brown;  }  </style>  </head>  <body>  <canvas id="dibujo" height="300" width="500"> |

|  |
| --- |
| Tu navegador no soporta &lt;canvas&gt;  </canvas>  <script>  var gráfico = document.getElementById("dibujo"); var texto = gráfico.getContext("2d");  texto.font = "40px Times New Roman"; texto.fillStyle = "blue";  texto.textAlign = "center";  texto.fillText("Mensaje", gráfico.width/2, gráfico.height/2);  </script>  </body>  </html> |
|  |

#### Imágenes

Para insertar una imagen en un elemento <canvas> se utiliza el siguiente método:

* + - * drawImage(imagen,izda,arriba): Dibuja la imagen “imagen” en la posición “izda”, “arriba”.



*No se puede dibujar una imagen antes de que dicha imagen se haya cargado por lo que hay que usar una función que se ejecute cuando se carga la página como puede ser* ***window.onload()***

**Ejemplo:**

|  |
| --- |
| <!DOCTYPE html>  <html lang="es-es"  <head>  <meta charset="UTF-8">  <title>Imágenes canvas</title>  <style>  body { background-color: #ccffcc; } #dibujo {  border:3px solid brown;  }  </style>  </head>  <body>  <h2>Imagen a usar en canvas</h2>  <img id="icono" src="torre.jpg" alt="Torre"><br>  <h2>Canvas</h2>  <canvas id="dibujo" height="300" width="500">  Tu navegador no soporta &lt;canvas&gt;  </canvas>  <script>  window.onload = function() {  var gráfico = document.getElementById("dibujo"); var imagen = gráfico.getContext("2d");  var img = document.getElementById("icono"); imagen.drawImage(img, 200, 50);  }  </script>  </body>  </html> |

|  |
| --- |
|  |