

Drawing text

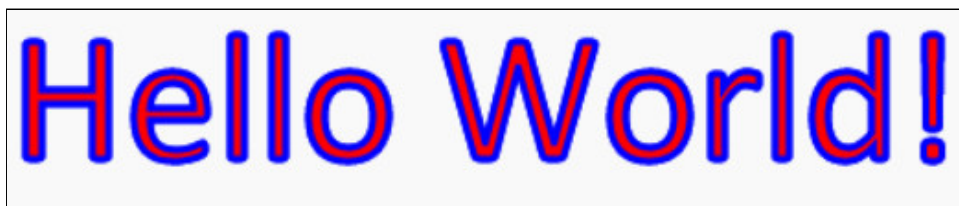
INTRODUCTION

The canvas API provides two main methods for drawing text: `ctx.strokeText(message, x, y)` and `ctx.fillText(message, x, y)`.

It also provides a set of context properties for setting the character font and style, for laying out the text, etc.

TYPICAL USE

Try this example online: <http://jsbin.com/cutoja/2/edit>



Source code extract:

```
context.font = "60pt Calibri";  
// .. set color, lineWidth, shadow etc.  
// 10, 10 is the start of the baseline, bottom of left leg of the "H" in the  
// "Hello World" example.  
context.fillText("Hello World!", 10, 10);  
// Or  
context.strokeText("Hello World!", 10, 10);
```

Look at the code from the example provided: <http://jsbin.com/cutoja/2/edit> - change the position where the text is drawn, change font attributes, etc.

THE CONTEXT.FONT PROPERTY:

It is possible to draw text in a canvas using the `font` property of the context to specify the font style (plain, bold, italic), the size, and the font name. Other properties such as `strokeStyle` or `fillStyle`, as well as other properties detailed in the next subchapters, will also be taken into account.

The font property is CSS-compliant, and accepts values like:

```
[font style][font weight][font size][font face]
```

Accepted values are:

- `font style`: normal, italic, oblique, inherit
- `font weight`: normal, bold, bolder, lighter, auto, inherit, 100, 200, 300, 400, 500, 600, 700, 800, 900
- `font size`: a size in pixels or in points, such as 60pt, 20px, 36px, etc.
- `font face`: Arial, Colibri, Times, Courier, etc. Some font faces may not work in all browsers.

Examples:

- `context.font = "60pt Calibri";`
- `context.font = "normal normal 20px Verdana";`
- `context.font = "normal 36px Arial";`
- `context.font = "italic bold 36px Arial";`

THE `FILLTEXT()` OR `STROKETEXT()` METHODS

The `fillText(message, x, y)` or `strokeText(message, x, y)` methods from the context will actually draw a text message at the origin of the baseline position. In the "Hello World" example, this is located at the bottom of the left leg of the "H".

There is a fourth optional parameter `maxWidth` that forces the text to fit into the given width, distorting it if necessary:

```
context.strokeText("Hello World!", x, y, [, maxWidth]);  
context.fillText("Hello World!", x, y, [, maxWidth]);
```

EXAMPLE THAT USES THE `MAXWIDTH` PARAMETER OF THE `STROKETEXT()` OR `FILLTEXT()` METHODS:

Try it online: <http://jsbin.com/zixeve/2/edit>



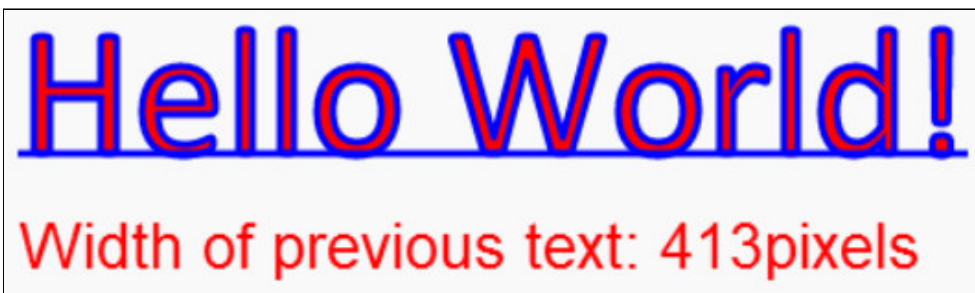
Hello World!
Hello World!
Hello World!

Source code extract:

```
...  
context.font = "60pt Calibri";  
context.lineWidth = 3;  
context.strokeStyle = "blue";  
context.fillStyle = "red";  
context.fillText("Hello World!", 10, 100);  
context.strokeText("Hello World!", 10, 100);  
10. // Draw text with constrained width of 250 pixels  
context.fillText("Hello World!", 10, 160, 250);  
context.strokeText("Hello World!", 10, 160, 250);  
// Constrain width to 150 pixels  
context.fillText("Hello World!", 10, 220, 150);  
context.strokeText("Hello World!", 10, 220, 150);
```

MEASURING THE WIDTH OF A GIVEN TEXT (BOUNDING BOX)

Try this example online: <http://jsbin.com/texigo/1/edit>



Hello World!

Width of previous text: 413pixels

The `ctx.measureWidth()` method can be used to get the current width in pixels of a given text, taking into account the diverse properties involved such as font, size, shadow, `lineWidth`,

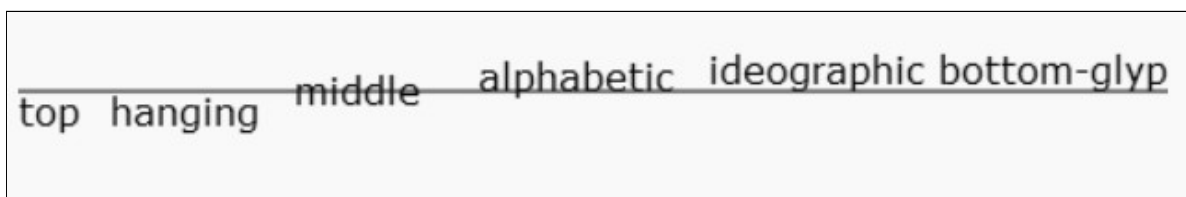
etc.

Source code extract from this example:

```
context.font = "60pt Calibri";
context.lineWidth = 3;
context.strokeStyle = "blue";
context.fillStyle = "red";
context.fillText("Hello World!", 10, 100);
context.strokeText("Hello World!", 10, 100);
var textMetrics = context.measureText("Hello World!");
10. var width = textMetrics.width;
    // Draw a text that displays the width of the previous drawn text
    context.font = "20pt Arial";
    context.fillText("Width of previous text: " + width + "pixels", 10, 150);
    // Draw the baseline of the given width
    context.moveTo(10, 100);
    context.lineTo(width+10, 100);
    context.stroke();
```

THE `CTX.BASELINE` PROPERTY: CHANGE THE WAY THE TEXT IS HORIZONTALLY DRAWN

Try this example online: <http://jsbin.com/mitugi/1/edit> (borrowed and adapted from <http://tutorials.jenkov.com/html5-canvas/text.html>)



The text baseline is important as it tells how the `y` parameter of the `fillText("some text", x, y)` and `strokeText("some text", x, y)` methods is interpreted.

The `baseline` property of the context is used to specify the different ways one can position the baseline of a given text. The example above shows the different possible values for this property and the corresponding results. The default value is "alphabetic" and corresponds to what has been used in the previous "Hello World" example.

Possible values:

Possible values for the baseline property	
top	The text is aligned based on the top of the tallest glyph in the text.
hanging	The text is aligned based on the line the text seems to hang from. This is almost identical to top, and in many cases, you cannot see the difference.
middle	The text is aligned according to the middle of the text.
alphabetic	The bottom of vertically oriented glyphs, e.g. western alphabet like the latin.
ideographic	The bottom of horizontally oriented glyphs.
bottom	The text is aligned based on the bottom of the glyph in the text, that extends furthest down in the text.

Typical use (taken from the example above):

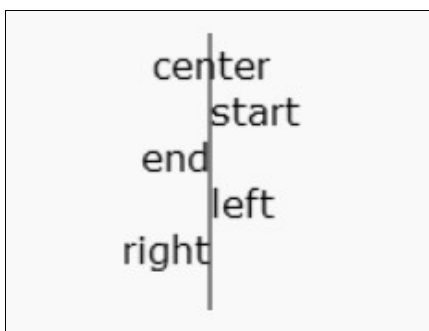
```

context.textBaseline = "top";
context.fillText("top", 0, 75);
context.textBaseline = "hanging";
context.fillText("hanging", 40, 75);
context.textBaseline = "middle";
context.fillText("middle", 120, 75);

```

HORIZONTAL TEXT ALIGNMENT

Try this example online: <http://jsbin.com/acudif/3/edit>



The `textAlign` property of the context tells how the `x` parameter will be used when calling `strokeText("some text", x, y)` and `fillText("some text", x, y)`. For example,

with `textAlign="center"`, the `x` parameter gives the position of the vertical center of the text, while in `textAlign="right"`, `x` corresponds to the rightmost position of the text.

Typical use (source code taken from the above example):

```
context.textAlign = "center";
context.fillText("center", 250, 20);
context.textAlign = "start";
context.fillText("start", 250, 40);
context.textAlign = "end";
context.fillText("end", 250, 60);
context.textAlign = "left";
context.fillText("left", 250, 80);
context.textAlign = "right";
10. context.fillText("right", 250, 100);
```

KNOWLEDGE CHECK 3.3.2 (NOT GRADED)

Which of these context properties are relevant to drawing text?

- ☐ ☐ `ctx.font`
- ☐ `ctx.size`
- ☐ `ctx.width`
- ☐ `ctx.textAlign`
- ☐ `ctx.textBaseline`
- ☐ `ctx.textOutlineWidth`

Note: Make sure you select all of the correct options. There are three correct ones!