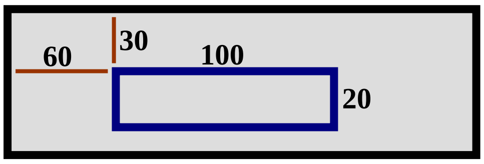
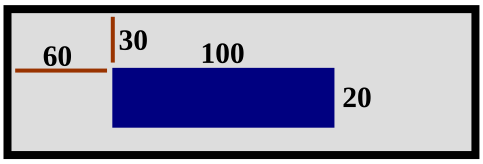
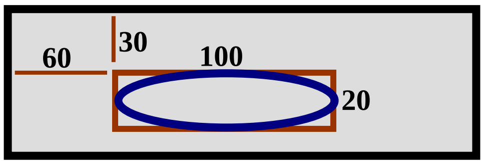
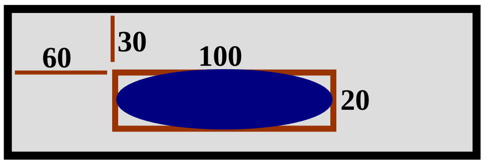
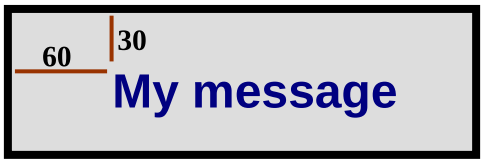
22 Jul 2009

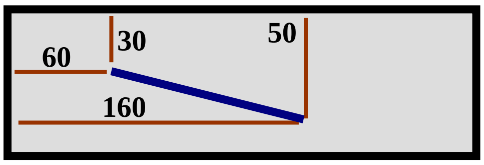
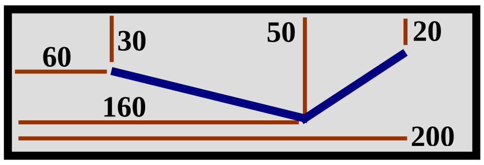
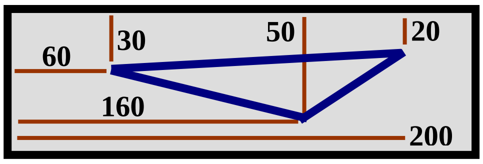
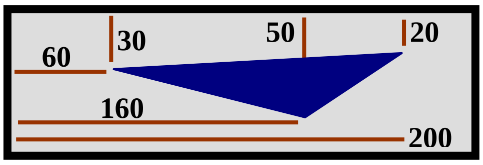
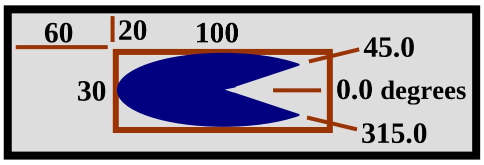
Have you ever wanted to draw a simple diagram or a basic illustration directly into a web page?
Something other than a stored image?
Something that displays quickly and looks crisp and sharp?
Something that allows for properties to be manipulated in code to produce animation effects?
Walter Zorn www.walterzorn.com has developed a very small (24 KB) library file which he generously makes available for download at no charge.
Javascript functions may be written into the HTML code of a web page to access the library and to draw vector graphical images directly into the page.
Walter's own site best demonstrates the speed, efficacy and manner of use of his work, but below is a listing of the functions I have made use of in my own site. Below that is a detailed explanation of how I have coded my pages.

Walter Zorn's Javascript Graphics Functions		

1.	Colour
	Stroke (line thickness)
		mycell.setColor("navy"); (OR – mycell.setColor("#000080");)
		mycell.setStroke(4);
		(Defaults for colour and stroke are black and 1.)

2.	Rectangle (outline)	mycell.drawRect(60, 30, 100, 20); 
3.	Rectangle (filled)	mycell.fillRect(60, 30, 100, 20); 
4.	Ellipse (outline)	mycell.drawEllipse(60, 30, 100, 20); 
5.	Ellipse (filled)	mycell.fillEllipse(60, 30, 100, 20); 

6.	Font
	String (text characters)
		mycell.setFont("Arial", "20px", "Font.BOLD");
		mycell.drawString("My message", 60, 30);
		

7.	Line (straight)	mycell.drawLine(60, 30, 160, 50); 
8.	Polyline	mycell.drawPolyline(new Array(60, 160, 200), new Array(30, 50, 20)); 
9.	Polygon (outline)	mycell.drawPolygon(new Array(60, 160, 200), new Array(30, 50, 20)); 
10.	Polygon (filled)	mycell.fillPolygon(new Array(60, 160, 200), new Array(30, 50, 20)); 
11.	Arc (filled)	mycell.fillArc(60, 20, 100, 30, 45.0, 315.0); 

12.	Printable
	Paint (on web page)
		mycell.setPrintable(true);
		mycell.paint();

STEP 1: Download Walter Zorn's library file, wz_jsgraphics.js from his website and save it in the same folder to be used to store the web page.

STEP 2: Use a text editor to write the HTML code for the shell of a web page and make reference in the page head to the stored library file.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD>
<TITLE>My Page</TITLE>
```

```
        <!--Reference Walter Zorn's DHTML javascript graphics library.//-->
<SCRIPT TYPE="text/javascript" SRC="wz_jsgraphics.js"></SCRIPT>
</HEAD>
<BODY>
</BODY>
</HTML>
```

STEP 3: In the page body, create a two-by-two table so as to be able to place the image inside a table cell, and then to be able to change the table properties and thus locate the image anywhere on the page.

STEP 4: In the fourth table cell, create a layer on which the image is to be displayed. The layer is declared with its unique identifier.

```
<BODY>
<TABLE BORDER=1>
<TR><TD WIDTH=300 HEIGHT=20><TD WIDTH=320>
<TR><TD HEIGHT=180><TD VALIGN=top>
        <!--Create layer in table cell for display of graphical image.//-->
<DIV ID="mycanvas" STYLE="position:relative; height:0px; width:0px"></DIV>
</TABLE>
</BODY>
```

STEP 5: In the same table cell, below creation of the layer, write Javascript code to declare a variable which is a new graphics object and which references the ID of the layer it is on.

STEP 6: Declare a new Javascript function.
Use the variable's name, together with instances of Walter Zorn's drawing functions, to build a composite image within the new Javascript function.

```
<DIV ID="mycanvas" STYLE="position:relative; height:0px; width:0px"></DIV>
<SCRIPT TYPE="text/javascript">
<!--
        /*Create function to define image objects on layer.*/
var myDraw = new jsGraphics("mycanvas");
function MyImage()
{
  myDraw.setColor("red");
  myDraw.fillEllipse(100, 0, 150, 100); //red ellipse
  myDraw.setColor("lime");
  myDraw.fillPolygon(new Array(220, 220, 290), new Array(30, 120, 120)); //lime triangles
  myDraw.fillPolygon(new Array(214, 214, 90), new Array(5, 120, 120));
  myDraw.setColor("blue");
  myDraw.fillRect(20, 146, 280, 20); //blue rectangle
  myDraw.setColor("magenta");
  myDraw.fillArc(100, 100, 200, 50, 180.0, 0.0); //magenta half ellipse
  myDraw.setStroke(3);
  myDraw.setColor("white");
  myDraw.drawLine(20, 150, 200, 150); //white line
  myDraw.setPrintable(true); //make image printable
  myDraw.paint(); //draw directly into the table cell's layer
}
//-->
</SCRIPT>
</TABLE>
```

STEP 7: Immediately after the Javascript function – which defines how the image is to be drawn – write a call to the function to implement drawing into the web page.

```
}
MyImage();
//-->
</SCRIPT>
```

STEP 8: Save the web page – possibly as MyPage.htm – then use a web browser to open it for viewing.
Admire, and perhaps edit, your handiwork (as shown below).