1 div wonder

what can you imagine with just one div and CSS?

Intent and focus is on exploring shapes in CSS.

Anubhav Saini

Theory

What are block elements or objects in HTML & CSS?

Any elements that can take shape of a box and have a line break before and after are known as block elements or objects.

What are width and height properties for?

They apply to block elements and make blocks as wide and high as specified by width and height values.

We will use pixels or percentages in this book.

Note: for more about these properties or values search internet, read w3c specifications or wait for the Hope : The Web Design Book, part 3.

What is border property?

For an element it specifies the borders on all four sides. It is shorthand for border-width, border-style and border-color. For example:

border: value-1 value-2 value-3;
means:
border-top: value-1 value-2 value-3;
border-right: value-1 value-2 value-3;
border-bottom: value-1 value-2 value-3;
border-left: value-1 value-2 value-3;
where:
border-[t|r|b|l]: value-1 value-2 value-3;
means:
border-[t|r|b|l]-width: value-1;
border-[t|r|b|l]-style: value-2;
border-[t|r|b|l]-color: value-3;
example:
border: 50px solid red;
sets creates red colored, solid, 50 pixels wide border on all four sides.

more details on border are beyond this book, you can read W3C specification, online blogs or wait for my books Hope 2 or 3 for it.

What is border-radius propery?

It specifies the radius of the border. If it is equal to the width of border and that element has 0 width and 0 height then it becomes a circle.

Negative radius is invalid.

border-radius: top-left top-right bottom-right bottom-left;

What is box-shadow propery?

Creates shadows for the box, shadows are identical to the box. You can create as many as you wish, a fact that we will find very useful.
Padding, margin, height and width of box do affect the shadow of the box.

What are -webkit- properties?

-webkit- is vendor specific prefix that is used for the properties that are experimental in one or other sense.

What is background-color propery?

It sets the background color for the box.
I would have told you that it is uninheritable but shines through children elements, but since this books is about one single div, inheritance is kind of out of it's reach and or goals.

What is transparent color?

It is any hue with 0 alpha channel value. I use RGBA (0, 0, 0, 0) for transparent color of which, transparent keyword is synonym
It is disparate from opacity.

Square

using height/width

code 1

```
<!DOCTYPE html>
<html>
 <head>
  <meta charset=”utf-8” />
  <title>code square</title>
  <style type=”text/css”>
            #protagonist{
                  background-color: Black;
                  height: 100px;
                  width: 100px;
            }
      </style>
 </head>
 <body>
      <div id=”protagonist”> </div>
 </body>
</html>
```

This is the last time we are seeing complete code in example, from now on we will deal with CSS code only.

It’s output will look like this:

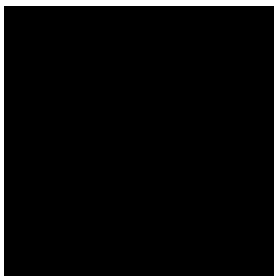


fig 1

using border

code 2

```
background-color: none;
width: 0px;
height: 0px;
border: 50px solid black;
```

It will generate same black box like shown in figure 1.

Rounded Square

We will use border squares from now on.

Add border-radius: 15px; to the code 2.

code 3

```
background-color: none;
width: 0px;
height: 0px;
border: 50px solid black;
border-radius: 15px;
```

This will curve the edges giving them a nice rounded border. A good visual treat in the world of plain old boxes.

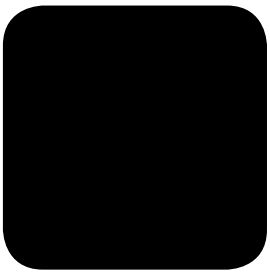


fig 2

If you continue increasing the radius of the browser, the more rounded it will become. Let’s give it 35 pixel radius:

```
border-radius:
```

Rectangle

Rectangle can be created using height and width properties. But unlike square we have to provide different values thus two types of rectangles are possible, wide and high. Also, we can use W/H and or define border to create a rectangle . Let's see wide then high in all 3 ways:

case 1.a wide rectangle using W/H

```
width: 200px;
height: 100px;
background-color: black;
```

case 1.b and using border and W/H

```
background-color: black;
width: 100px;
height: 0px;
border: 50px solid black;
```

case 1.c again using only border

```
background-color: none;
width: 0px;
height: 0px;
border-width: 50px 100px;
border-style: solid;
border-color: black;
```

All of the above will create same output as follows:

We can acthieve high rectangle by flipping values in all three cases above.

case 2.a high rectangle using W/H

```
width: 100px;
height: 200px;
background-color: black;
```

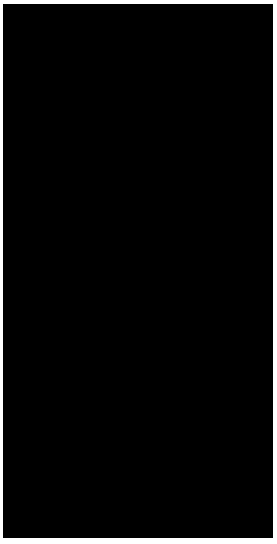
case 2.b and using border and W/H

```
background-color: black;
width: 00px;
height: 100px;
border: 50px solid black;
```

case 2.c again using only border

```
background-color: none;
width: 0px;
height: 0px;
border-width: 100px 50px;
border-style: solid;
border-color: black;
```

All of the above will create same output as follows:

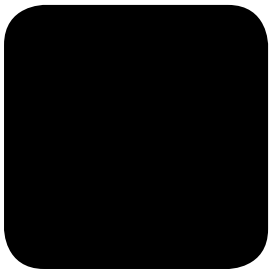


Smooth Edge Square Circle

We will take code from square example #2 and add border-radius: 15px; to CSS rules.

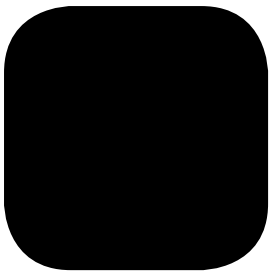
```
#protagonist{
        background-color: none;
        width: 0px;
        height: 0px;
        border: 50px solid black;
        border-radius: 15px;
}
```

This will smoothen the edges of border. This will not work with square example #1, because there is no border to modify at all in that example.

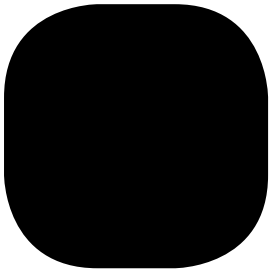


We can increase radius to 25 pixels to make it a bit more rounded.

```
border-radius: 25px;
```



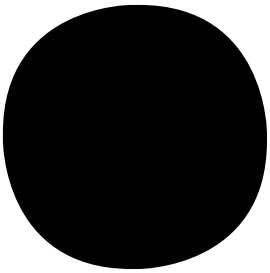
```
border-radius: 35px;
```



which brings us to:

If we take border radius too far upto the point where it is equal to the border width, we will end up creating a circle.

```
#protagonist{
        background-color: none;
        width: 0px;
        height: 0px;
        border: 50px solid black;
        border-radius: 50px;
}
```



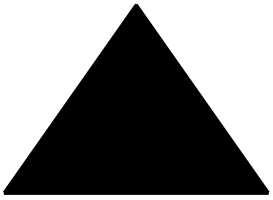
Isosceles triangle

Isosceles triangles are those with two sides same, with this in mind, we will create four triangles that will be pointing different directions namely top, right, bottom and left. A triangle in our book faces top when it's tip is in that direction. Question is which tip? Obvious answer would be one that is made up of two equal in length sides.

Pay close attention, we are about to blow up the border shorthand notation.

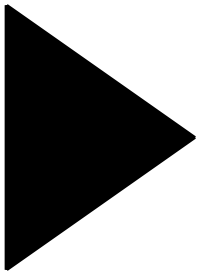
```
#protagonist{
        height: 0px;
        width: 0px;
        border-style: solid;
        border-color: transparent transparent black transparent;
        border-width: 0px 25px 50px 25px;
}
```

And it will look like:



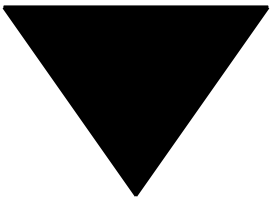
```
border-width: 25px 0 25px 50px;
```

will generate right isosceles triangle:



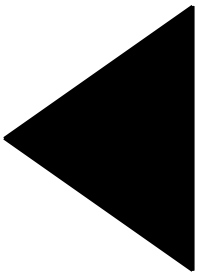
```
border-width: 50px 25px 0 25px;
```

will generate a down ward triangle:



```
border-width: 25px 50px 25px 0;
```

will generate a left facing triangle:



Right triangle

Right angled triangles look very good when put in corner of an HTMl document.

This time, we won't call them top, right, up or down triangles for the sake of clarity and unambiguity.

We will call them top-left, top-right, bottom-right and bottom-left, depending upon which direction their base and perpendicular are. Also, watch closely two properties border-color and border-width. I'll show you two ways you can do every right triangle.

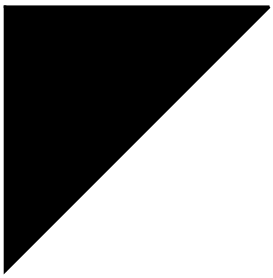
case 1.a with 100px width

```
#protagonist{
	border-style: solid;
	border-color: transparent transparent transparent Black;
	border-width: 0px 0px 100px 100px;
}
```

case 1.b with 50px width

```
#protagonist{
	border-style: solid;
	border-color: Black transparent transparent Black;
	border-width: 50px;
}
```

output in both cases will be same, pixel by pixel:



Don't want to lie to you or something, these right triangles are isosceles triangles too. :)

case 2.a top-right triangle

```
	border-color: transparent Black transparent transparent;
	border-width: 0px 100px 100px 00px;
```

case 2.b

```
	border-color: Black Black transparent transparent;
	border-width: 50px;
```

result is still identical:

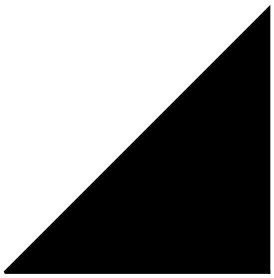
case 3.a bottom-right triangle

```
	border-color: transparent Black transparent transparent;
	border-width: 100px 100px 00px 00px;
```

case 2.b

```
	border-color: transparent Black Black transparent;
	border-width: 50px;
```

result is as expected:

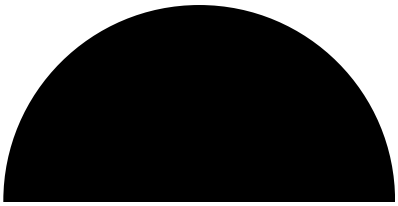


You caught the drift, right! So, as an exercise, try the remaining one.

Semi circle

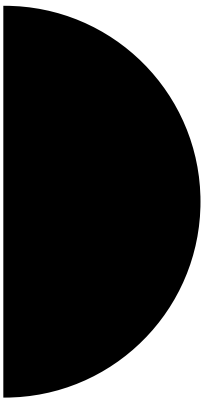
We have created circle using border radiuus, quite similarly semi circle can be achieved. Of course there are four semi circles top, right, bottom and left.

```
width: 0px;
border-style: solid;
border-color: Black Black transparent Black;
border-width: 50px 50px 0 50px;
border-radius: 50px 50px 0 0;
```



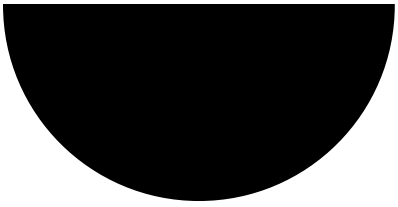
for right semi circle

```
border-color: Black Black Black transparent;
border-width: 50px 50px 50px 0;
border-radius: 0 50px 50px 0;
```



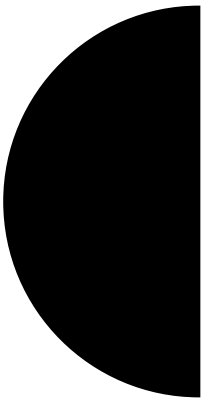
for bottom semi circle

```
border-color: transparent Black Black;
border-width: 0 50px 50px 50px;
border-radius: 0 0 50px 50px;
```



left semi circle:

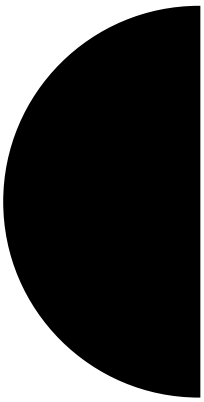
```
border-color: Black transparent Black Black;
border-width: 50px 0 50px 50px;
border-radius: 50px 0 0 50px;
```



This all was predictable, but here's the shocker for left semi circle, take right semi circle's color and width and above left's radius:

```
border-color: Black Black Black transparent;
border-width: 50px 50px 50px 0;
border-radius: 50px 0 0 50px;
```

same result:

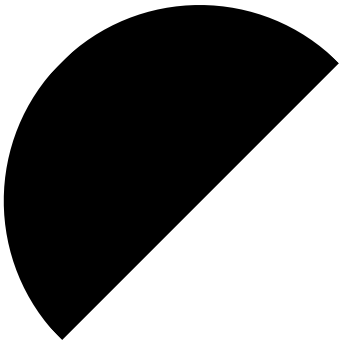


There are few more semi circles left, like top-left, top-right, bottom-right, bottom-left.

We can use transform CSS property, but let's just play with properties we know yet.

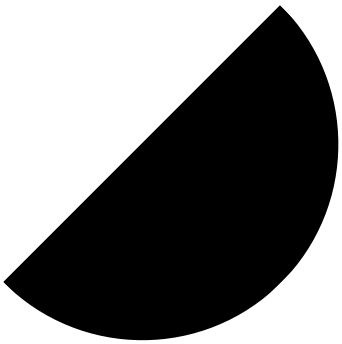
top-left semi circle

width: 0px;
border-style: solid;
border-color: Black transparent transparent Black;
border-width: 100px;
border-radius: 100px 100px 00px 100px;



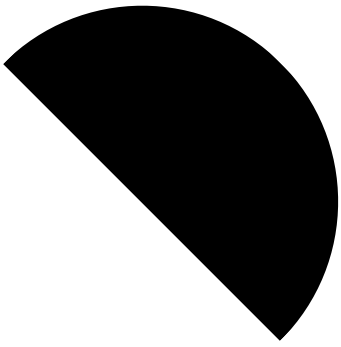
bottom-right semi circle

border-color: transparent Black Black transparent;
border-radius: 00px 100px 100px 100px;



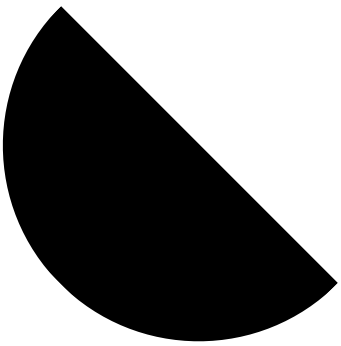
top-right semi circle

border-color: Black Black transparent transparent;
border-width: 100px;
border-radius: 100px 100px 100px 00px;



bottom-left semi circle

border-color: transparent transparent Black Black;
border-radius: 100px 00px 100px 100px;



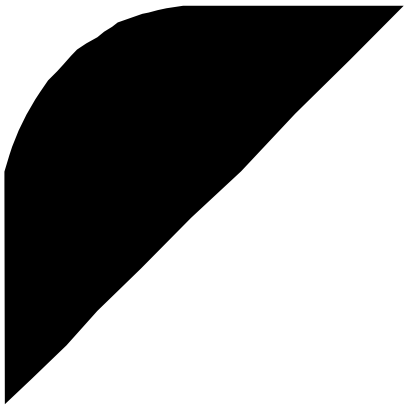
U.F.O / cool Hat

It reminds me of some Star Wars character who used to wear black. I am sorry I don't know much about Star Wars or Star Trek for that matter.

We will first deal with top-left, top-right, bottom-right, and bottom-left; later we will access top, right, bottom and left ones.

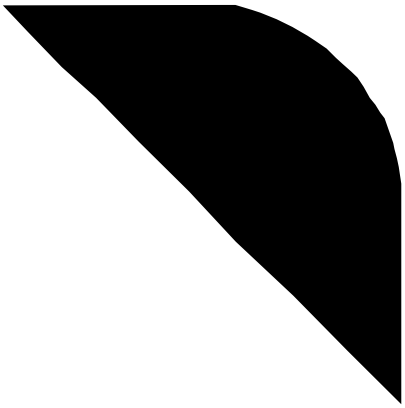
top-left UFO or star wars hat.

```
width: 0px;
border-style: solid;
border-color: Black transparent transparent Black;
border-width: 100px;
border-radius: 100px 00px;
```



top-right U.F.O

```
border-color: Black Black transparent transparent;
border-width: 100px;
border-radius: 00px 100px ;
```



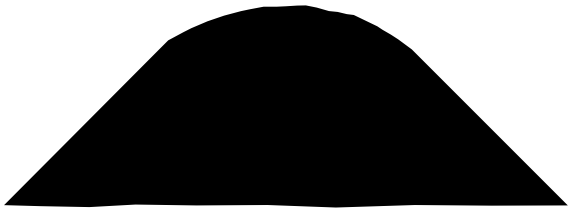
Other two are guessable. You must have found pattern to it by now. So, let's rotate the things a bit.

We can not create top, right, bottom or left UFO this way, so what we will do is rotate them 45 degrees. top-left on 45 degree rotation will yield top UFO. Rest is what you can play with.

Let's take top-left and create a top UFO

```
border-color: Black transparent transparent Black;
border-width: 100px;
border-radius: 100px 00px;
-webkit-transform: rotate(45deg);
margin: 0 35px;
```

margin is required because in simplest cases it will go to left and get clipped.



Other are also possible. Play with 45 degrees of increment in rotation.

Since we now know about rotate, let's visit every example we have created till now.

Rotate Examples

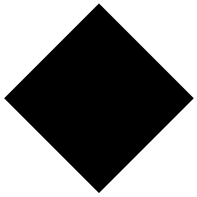
Square is first one. On rotating it becomes diamond from playing cards, may be by a bit of width/height manipulation.

This was our first square:

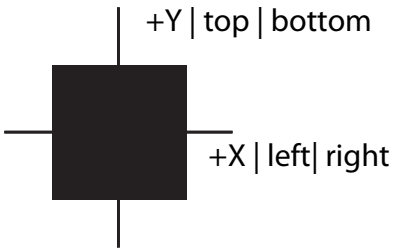
```
background-color: Black;
height: 100px;
width: 100px;
```

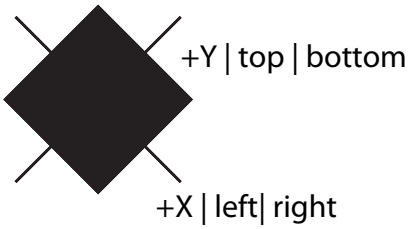
Let's rotate it 45 degrees:

```
-webkit-transform:rotate(45deg);
```



Positive degrees of rotation in CSS are clockwise, so if you look at the axis of the both squares they will look like this:





Rotating rectangle is same thing, rotating circle is pointless unless it was multicolored, rotating semi circle is similar to rotating square, rotating triangles is same old boring till now.

Hmm, seem like we have done everything we could do with just one div.

Okay, so just read more theory and go home. Shall we?

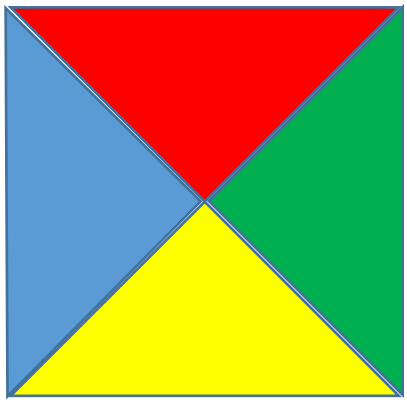
More Theory

How does border generates triangles?

Let's see a bordered div under microscope, er... a coloroscope. Whatever.

```
#protagonist{
        border: 50px solid;
        border-color: red green yellow blue;
}
```

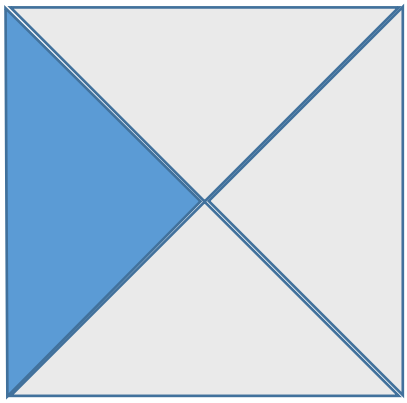
This will generate something like this:



So, when you turn any border side color to transparent, you miss that triangle. For example, to make only the blue triangle, all I need to do is:

```
border-color: transparent transparent transparent blue;
```

and I would get something like below (assume transparent is not quite transparent [:-)])

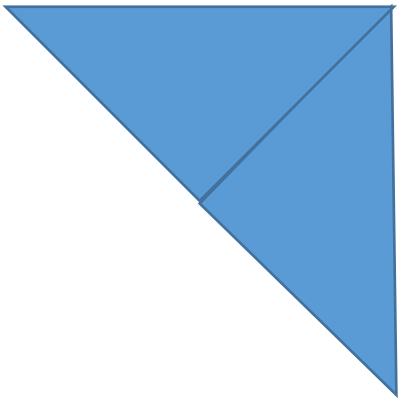


Cool, so now you can create triangles, but what about right angled triangles, where did they come from?

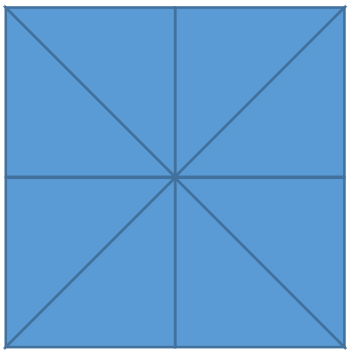
There were two type of right angled triangles, one is quite easy to figure out now, all you have to do is make two border sides transparent. Let's see:

```
border-color: Blue Blue transparent transparent;
```

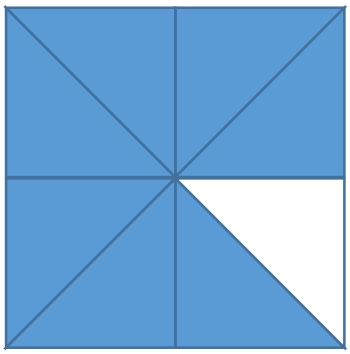
will produce something like this:



The other way is quite the convoluted one. Let's bring up the wireframe.



So, now say I want right top right angle triangle, I would require to isolate the white one in the image next page.



For that, we need to understand what happens when border width is omitted for a side?

When we omit width for one side, we completely remove that side border, thus

border-width: 50px 0 50px 50px;

will become:

now, if I need only the red top right triangle, I need to either remove bottom border too and make left border transparent or make bottom and left border transparent. Let's do former.

border-width: 50px 0 0 50px;
border-color: red transparent;

Note that I have set top and bottom border red, but since I have also made width of bottom border zero, it won't show.

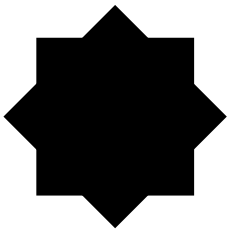
Output of such would be:

Various shapes

8 corners

```
#protagonist{
        margin: 100px;
        height: 0px;
        width: 0px;
        border: 50px solid Black;
}
#protagonist::after{
        display: block;
        content: “”;
        height: 0;
        width: 0;
        border: 50px solid Black;
        -webkit-transform: rotate(45deg);
        position: relative;
        top: -50px;
        left: -50px;
}
```

will generate image like this:



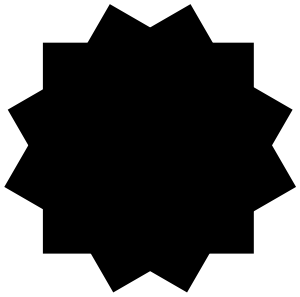
What we have done is that create a box using #protagonist and then create another box rotated 45 deg CC using #protagonist::after.

Then using relative positioning overlapped them.

12 corners

Important thing here is that I rotated ::after and ::before psuedo boxes 30 and -30 degrees.

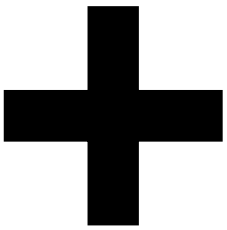
It looks like this:



```
#protagonist{
        margin: 100px;
        height: 0px;
        width: 0px;
        border: 50px solid Black;
}

#protagonist::after{
        display: block;
        content: “”;
        height: 0;
        width: 0;
        border: 50px solid Black;
        -webkit-transform: rotate(-30deg);
        position: relative;
        top: -150px;
        left: -50px;
}

#protagonist::before{
        display: block;
        content: “”;
        height: 0;
        width: 0;
        border: 50px solid Black;
        -webkit-transform: rotate(30deg);
        position: relative;
        top: -50px;
        left: -50px;
}
```

Sign of Addition

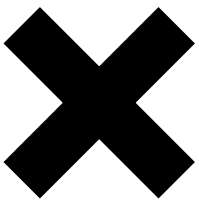
```
#protagonist {
        margin: 10px 50px;
        height: 0px;
        width: 0px;
        border-style: solid;
        border-color: black;
        border-width: 50px 10px;
}

#protagonist::after {
        display: block;
        content: “”;
        border-style: solid;
        border-color: black;
        border-width: 10px 50px;
        top: -10px;
        left: -50px;
        position: relative;
}
```



Creating sign of subtraction is even more trivial.

Sign of Multiplication

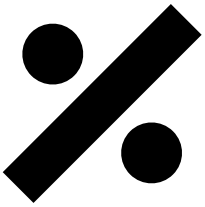
```
#protagonist {
        margin: 10px 50px;
        height: 0px;
        width: 0px;
        border-style: solid;
        border-color: black;
        border-width: 50px 10px;
        -webkit-transform: rotate(45deg);
}

#protagonist::after {
        display: block;
        content: “”;
        border-style: solid;
        border-color: black;
        border-width: 10px 50px;
        top: -10px;
        left: -50px;
        position: relative;
}
```



Creating sign of division is even more trivial.

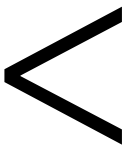
Sign of modulus or percentage

```
#protagonist {
	margin: 75px;
	height: 0px;
	width: 0px;
	border-style: solid;
	border-color: black;
	border-width: 10px 100px;
	-webkit-transform: rotate(-45deg);
}

#protagonist::before {
	display: block;
	content: “”;
	border-style: solid;
	border-color: black;
	border-width: 20px;
	top: -75px;
	left: -25px;
	position: relative;
	-webkit-transform: rotate(-90deg);
	border-radius: 20px;
}

#protagonist::after {
	display: block;
	content: “”;
	border-style: solid;
	border-color: black;
	border-width: 20px;
	top: -10px;
	left: -25px;
	position: relative;
	-webkit-transform: rotate(-90deg);
	border-radius: 20px;
}
```



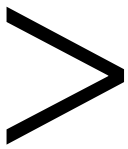
Sign of less than

```
#protagonist{
	margin: 100px;
	height: 0px;
	width: 0px;
	border-width: 10px 100px;
	border-style: solid;
	-webkit-transform: rotate(-35deg);
}

#protagonist::after{
	display: block;
	content: “”;
	height: 0;
	width: 0;
	border-width: 10px 100px;
	border-style: solid;
	-webkit-transform: rotate(70deg);
	position: relative;
	top: 77px;
	left: -175px;
}
```



Sign of greater than

```
#protagonist{
        margin: 100px;
        height: 0px;
        width: 0px;
        border-width: 10px 100px;
        border-style: solid;
        -webkit-transform: rotate(35deg);
}

#protagonist::after{
        display: block;
        content: “”;
        height: 0;
        width: 0;
        border-width: 10px 100px;
        border-style: solid;
        -webkit-transform: rotate(-70deg);
        position: relative;
        top: 77px;
        left: -24px;
}
```



Sign of exclamation

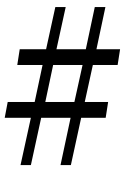
```
#protagonist{
        margin: 100px;
        height: 0px;
        width: 0px;
        border-width: 100px 10px;
        border-style: solid;
}

#protagonist::after{
        display: block;
        content: “”;
        height: 0;
        width: 0;
        border-width: 10px;
        border-style: solid;
        position: relative;
        top: 110px;
        left: -10px;
}
```

It will create a rectangualar/square-ish exclamation mark. Following is kind of medal-ish and sexy.

```
#protagonist {
        height: 0px;
        width: 0px;
        border-style: solid;
        border-color: black;
        border-width: 100px 60px;
        position: relative;
        left: 100px;
        border-radius: 0 0 10px 10px;
}

#protagonist::after {
        display: block;
        content: “”;
        border-style: solid;
        border-color: black;
        border-width: 40px 50px;
        top: 110px;
        left: -50px;
        position: relative;
        border-radius: 10px 10px 40px 40px;
}
```

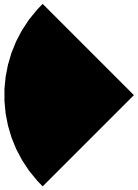
Sign of hash

```
#protagonist {
        margin: 100px;
        height: 0px;
        width: 0px;
        border-style: solid;
        border-color: black;
        border-width: 100px 10px;
        -webkit-transform: skew(-10deg);
        box-shadow: 90px 0 0 0 black;
}

#protagonist::after {
        display: block;
        content: “”;
        border-style: solid;
        border-color: black;
        border-width: 10px 100px;
        top: -50px;
        left: -50px;
        position: relative;
        -webkit-transform: skew(-10deg);
        box-shadow: 10px 90px 0 0 black;
}
```



Quarter circle and pizza slice

```
#protagonist{
        border-style: solid;
        border-color:  transparent transparent
transparent Black;
        border-width:  100px  00px 100px  100px;
        border-radius: 100px 00px 00px 100px;
}
```

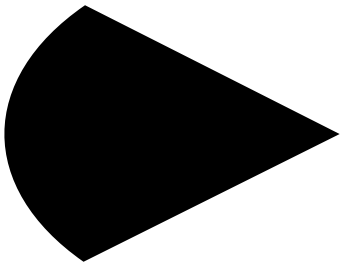


converting it into other circles is quite the same drill we have seen with earlier examples. All you have to do is shuffle the values around.

converting it into a pizza slice requires increasing border width of the side bulging out. for above example we need to change border width as follows:

```
        border-width:  100px  00px 100px  170px;
```

you can make your slice as big as you want. For me 170 pixels looked nice enough.



Trapezoid

Do you know that Americans call trapezoid: A quadrilateral with two parallel sides where as British call trapezoid: A quadrilateral with no parallel sides.

Anyway, remember that top isosceles triangle a while ago, we are going to create trapezoid using that same triangle.

This code creates upward triangle:

```
#protagonist{
        height: 0px;
        width: 0px;
        border-style: solid;
        border-color: transparent transparent Black transparent;
        border-width: 0px 25px 50px 25px;
}
```

And this creates an upward trapezoid:

```
#protagonist{
        height: 0px;
        width: 50px;
        border-style: solid;
        border-color: transparent transparent Black transparent;
        border-width: 0px 25px 50px 25px;
}
```

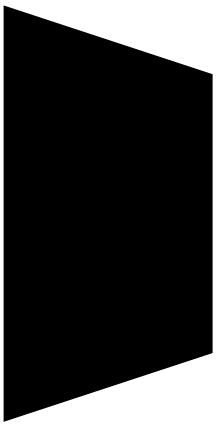
It looks like this:

This code creates a rightward triangle:

```
#protagonist{
        height: 0px;
        width: 0px;
        border-style: solid;
        border-color: transparent transparent transparent Black;
        border-width: 25px 0px 25px 50px;
}
```

to convert it into a trapezoid we need height:

```
#protagonist{
        height: 50px;
        width: 0px;
        border-style: solid;
        border-color: transparent transparent transparent Black;
        border-width: 25px 0px 25px 50px;
}
```



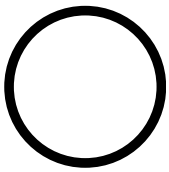
Now, since this basic concept is clear, we can create others easily.

Ring

Creating ring is application of height and width on a bordered circle. Also, the background color has to be applied so that the border, outer shell, and content region, cavity, can be differentiated visually.

Alas, there are no top, right, bottom or left rings. We have to deal with one. but don't be sad, next up we will be creating a diamond. Soon enough we will have a diamond ring. :)

```
#protagonist{
        height: 50px;
        width: 50px;
        border: 5px solid Black;
        border-radius: 30px;
}
```



But isn't a single ring boring? What about two of them? (what's emoticon for showing teeth and growing horns?)

```
#protagonist{
        height: 50px;
        width: 50px;
        border: 5px solid Black;
        border-radius: 30px;
        position: relative;
        top: 30px;
        left: 30px;
}

#protagonist::after{
        display: block;
        content: '';
        height: 70px;
        width: 70px;
        border: 5px solid Silver;
        border-radius: 40px;
        position: relative;
        top: -15px;
        left: -15px;
}
```

BELIEVE IT OR NOT, ADOBE INDESIGN CS 5.5 CAN NOT HANDLE RING IN RING AT ALL.

Can we create Olympics rings? Sadly No. Not that I can think of right now. sigh!

But, we can create 3/5th of Olympic rings, let's keep southern continents out of this personal Olympics.

#protagonist is same, changes are in position of ::after and ::before got what was ::after's in earlier code.

```
#protagonist::after{
        display: block;
        content: "";
        height: 50px;
        width: 50px;
        border: 5px solid Black;
        border-radius: 30px;
        position: relative;
        top: -65px;
        left: 76px;
}

#protagonist::before{
        display: block;
        content: "";
        height: 50px;
        width: 50px;
        border: 5px solid Black;
        border-radius: 30px;
        position: relative;
        top: -5px;
        left: 36px;
}
```

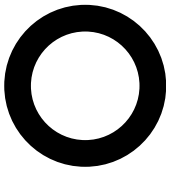
It creates three rings in one straight line, if you wish you can play with ::before's top value and put them in zigzag fashion like:

BELIEVE IT OR NOT, ADOBE INDESIGN CS 5.5 CAN NOT HANDLE RING IN RING AT ALL.

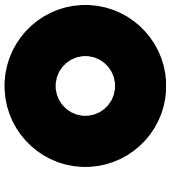
actually, you will need to play with top and left of both ::after and ::before.

Since we can maipulate width of the border and dimensions of content region,why don't we create heavier rings or tyres. You can imaging them to be rubber tyres.

Increase border width and compensate radius, something like this will emerge.



Though if you go for wider ring, it will look like a (I don't know what it is called in English, but it is used in plumbing and in some screw fastening mechanisms) :)



A bit more and it will become a cd, increase size and it will become old gramophone disk:

Some more things to try can be:

BELIEVE IT OR NOT, ADOBE INDESIGN CS 5.5 CAN NOT HANDLE RING IN RING AT ALL.

BELIEVE IT OR NOT, ADOBE INDESIGN CS 5.5 CAN NOT HANDLE RING IN RING AT ALL.

BELIEVE IT OR NOT, ADOBE INDESIGN CS 5.5 CAN NOT HANDLE RING IN RING AT ALL.

BELIEVE IT OR NOT, ADOBE INDESIGN CS 5.5 CAN NOT HANDLE RING IN RING AT ALL.

Parallelogram

Remember something like a quadrilateral whose opposite sides are both parallel and are equal in length, such a beauty worldwide is known as a parallelogram.

There are more than one way for creating a parallelogram, let's start from beginning.

from square

draw a square and add this to CSS declarations:

```
-webkit-transform: skew(30deg);
```

voila!

still code is here:

```
#protagonist{
    background-color: Black;
    height: 100px;
    width: 100px;
    -webkit-transform: skew(-30deg);
}
```

which generates a parallelogram like this:

of course, skew skews objects, negative angles are CW and positive are CCW.

TIP:
You can also deploy rotate(angle) and translate(length) along with skew(angle).

from triangles

create a (say) upward triangle. Spawn a ::after element, turn it into a downward triangle; and you are done.

```
#protagonist{
    margin: 100px;
    height: 0px;
    width: 0px;
    border-style: solid;
    border-color: transparent transparent black transparent;
    border-width: 0 25px 40px 25px;

}

#protagonist::after{
    display: block;
    content: "";
    height: 0px;
    width: 0px;
    border-style: solid;
    border-color: black transparent  transparent transparent;
    border-width: 40px 25px 0 25px;

}
```

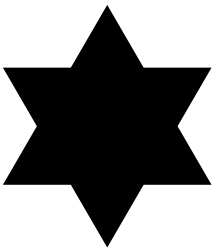
TIP
Just an FYI, if you don't spawn ::after, and in the #protagonist code add -webkit-box-reflect: below; you will get a diamond.

Stars

Remember triangles? same thing but now we will create and use multiple triangles to create various stars.

pointy 6 star

```
#protagonist{
        height: 0px;
        width: 0px;
        border-style: solid;
        border-color: transparent transparent Black transparent;
        border-width: 0px 25px 50px 25px;
}

#protagonist::before{
        display: block;
        content: “”;
        height: 0px;
        width: 0px;
        border-style: solid;
        border-color: Black transparent transparent transparent;
        border-width: 50px 25px 0px 25px;
        position: relative;
        left: -25px;
        top: 15px;
}
```



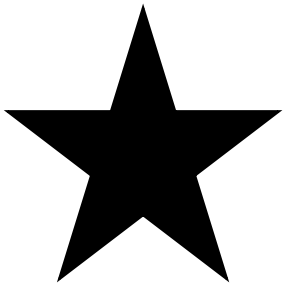
Basically these are just two stars on top of each other, one of them is upside down.

Creating pointy 5 star proved to be a mild challenge. I wanted to use box reflection. Talk about overkill! But, just couldn't get handle on the approach angle. Ended up using same old triangles in same old way.

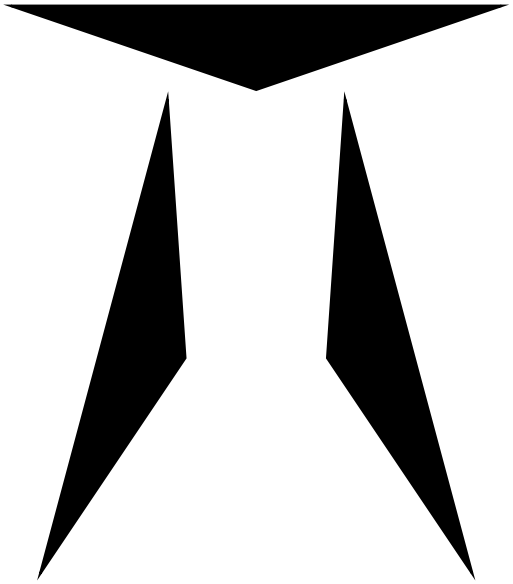
Pointy 5 doesn't look that cool, may be because all the triangles are identical. In another attempt I will use different triangles, a recipe which I cooked a while ago.

pointy 5 star

```
#protagonist{
        margin: 100px;
        height: 0px;
        width: 0px;
        border-style: solid;
        border-color: black transparent transparent transparent;
        border-width: 50px 100px 0 100px;
}

#protagonist::before{
        display: block;
        content: “”;
        height: 0px;
        width: 0px;
        border-style: solid;
        border-color: transparent black transparent transparent;
        border-width: 100px 50px 100px 0;
        -webkit-transform: rotate(-25deg);
        position: relative;
        left: 0px;
        top: -110px;
}

#protagonist::after{
        display: block;
        content: “”;
        height: 0px;
        width: 0px;
        border-style: solid;
        border-color:  transparent transparent transparent black;
        border-width:  100px 0 100px 50px;
        -webkit-transform: rotate(25deg);
        position: relative;
        left: -40px;
        top: -310px;
}
```



5 pointy star is created using 3 identical triangles, rotated and painstakingly positioned to give illusion of a star.



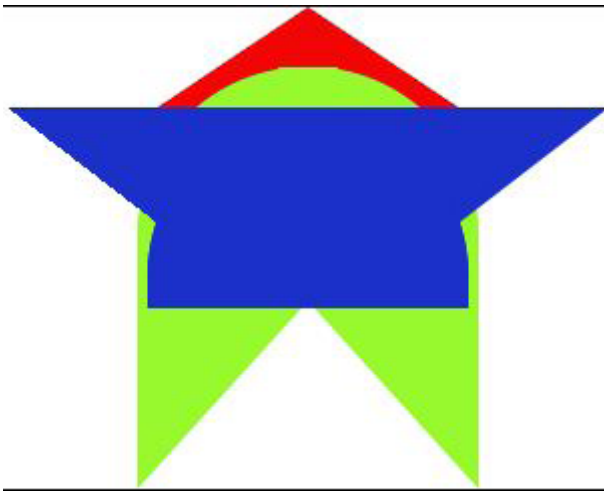
There is another way where we can use an upward triangle and those UFOs, except now they will be a bit distorted; er... actually a lot distorted. Also box-reflect property is used. It is one ugly looking star.

```
#protagonist{
        margin:199px;
        height: 0px;
        width: 0px;
        border-style: solid;
        border-color: transparent transparent Black
transparent;
        border-width: 0px 75px 50px 75px;
}

#protagonist::after{
        display: block;
        content: “”;
        width: 0px;
        border-style: solid;
        border-color: black black transparent
transparent;
        border-width: 100px 100px 0px 130px;
        position: relative;
        border-radius: 00px 84px;
        top: -50px;
        left: -150px;
        -webkit-box-reflect: right -160px;
}
```

```
#protagonist::before{
        display: block;
        content: “”;
        width: 0px;
        border-style: solid;
        border-color: black black transparent
transparent;
        border-width: 100px 100px 0px 110px;
        position: relative;
        border-radius: 00px 84px;
        top: 85px;
        left: -140px;

        -webkit-transform: rotate(-90deg);
        -webkit-box-reflect: above -170px;
}
```

t

What I have done really doesn’t make sense. So, I will create pointy 5 star in a bit more clean way. This is how it breaks down to:



pointy 5 cleaner

Star that you saw on the last page wasn't a star. A 5 point star looks like pretty much this:

After many many hit and trials (3 to be exact) I put together three triangles and created it. It's not exactly a star that I had in mind but yeah. Whatever.

```
#protagonist{
	margin: 100px;
	height: 0px;
	width: 0px;
	border-style: solid;
	border-color: transparent transparent black transparent;
	border-width: 0 25px 40px 25px;

}

#protagonist::before{
	display: block;
	content: "";
	height: 0px;
	width: 0px;
	border-style: solid;
	border-color: transparent black transparent transparent;
	border-width: 94px 44px 50px 0;
	-webkit-transform: rotate(67deg);
	position: relative;
	left: -15px;
	top: -21px;
}

#protagonist::after{
	display: block;
	content: "";
	height: 0px;
	width: 0px;
	border-style: solid;
	border-color: transparent black transparent transparent;
	border-width: 50px 40px 94px 0;
	-webkit-transform: rotate(113deg);
	position: relative;
	left: -23px;
	top: -167px;
}
```

With that I refuse to further investigate stars. Let's learn how to draw English capital letters via CSS.