HTML5& CSS3

Presentation by Uplatz

Contact Us: https://training.uplatz.com/
Email: info@uplatz.com
Phone:+44 7836 212635

Uplatz

Table Of Contents:

Grid:

- Grid layout is a new and powerful CSS layout system that allows to divide a web page content into rows and columns in an easy way.

Basic Example:

Property Possible Values

display grid / inline-grid

- The CSS Grid is defined as a display property.
- It applies to a parent element and its immediate children only.

Consider the following markup:

```
<section class="container">
 <div class="item1">item1</div>
 <div class="item2">item2</div>
```

```
<div class="item3">item3</div>
 <div class="item4">item4</div>
</section>
```

- The easiest way to define the markup structure above as a grid is to simply set its display property to grid:

```
.container {
 display: grid;
}
```

- However, doing this will invariably cause all the child elements to collapse on top of one another.
- This is because the children do not currently know how to position themselves within the grid. But we can explicitly tell them.

First we need to tell the grid element .container how many rows and columns will make up its structure and

we can do this using the grid-columns and grid-rows properties (note the pluralisation):

```
.container {
 display: grid;
 grid-columns: 50px 50px 50px;
 grid-rows: 50px 50px;
}
```

- However, that still doesn't help us much because we need to give an order to each child element.
- We can do this by specifying the grid-row and grid-column values which will tell it where it sits in the grid:

```
.container .item1 {
 grid-column: 1;
 grid-row: 1;
}
```

```
.container .item2 {
 grid-column: 2;
 grid-row: 1;
}
.container .item3 {
 grid-column: 1;
 grid-row: 2;
}
.container .item4 {
 grid-column: 2;
  grid-row: 2;
}
```

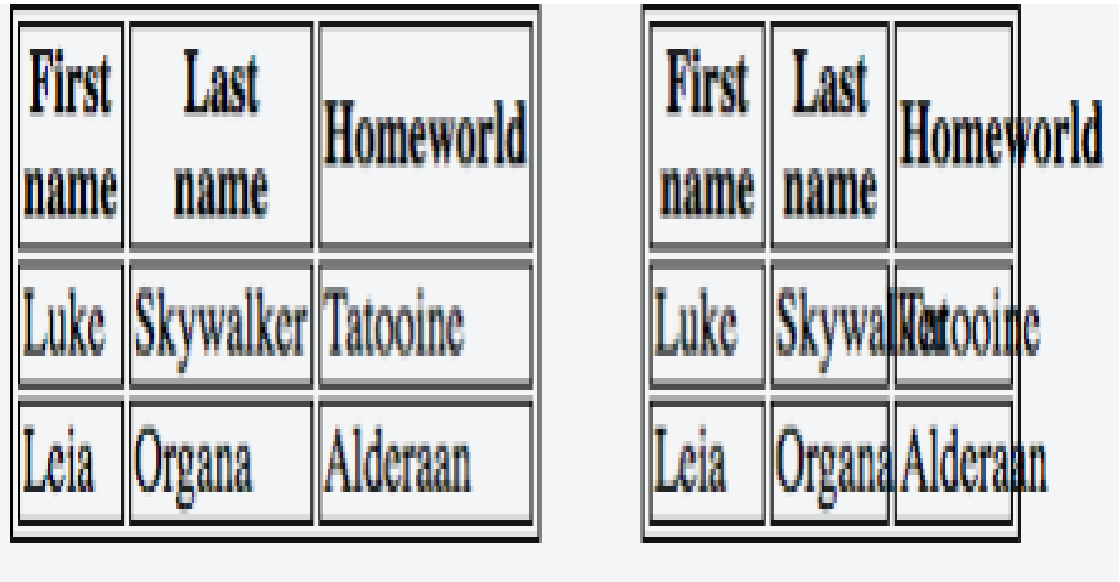
- By giving each item a column and row value it identifies the items order within the container

Tables

table-layout:

The table-layout property changes the algorithm that is used for the layout of a table.

Below an example of two tables both set to width: 150px:



First name	Last name	Homeworld
Luke	Skywalker	Tatooine
Leia	Organa	Alderaan

First name	Last name	Homeworld
Luke	Skywalker	Tatooine
Leia	Organa	Alderaan

- The table on the left has table-layout: auto while the one on the right has table-layout: fixed.
- The former is wider than the specified width (210px instead of 150px) but the contents fit.
- The latter takes the defined width of 150px, regardless if the contents overflow or not.

Value **Description**

auto This is the default value. It defines the layout of the table to be determined by the contents of its' cells.

fixed This value sets the table layout to be determined by the width property provided to the table. If the content of a cell exceeds this width, the cell will not resize but instead, let the content overflow.

empty-cells

- The empty-cells property determines if cells with no content should be displayed or not.
- This has no effect unless border-collapse is set to separate.
- Below an example with two tables with different values set to the empty-cells property:

First name	Last name	Homeworld
Luke		Tatooine
Leia	Organa	

First name	Last name	Homeworld
Luke		Tatooine
Leia	Organa	

- The table on the left has empty-cells: show while the one on the right has empty-cells: hide.
- The former does display the empty cells whereas the latter does not.

Value Description

show This is the default value. It shows cells even if they are empty.

hide This value hides a cell altogether if there are no contents in the cell.

border-collapse

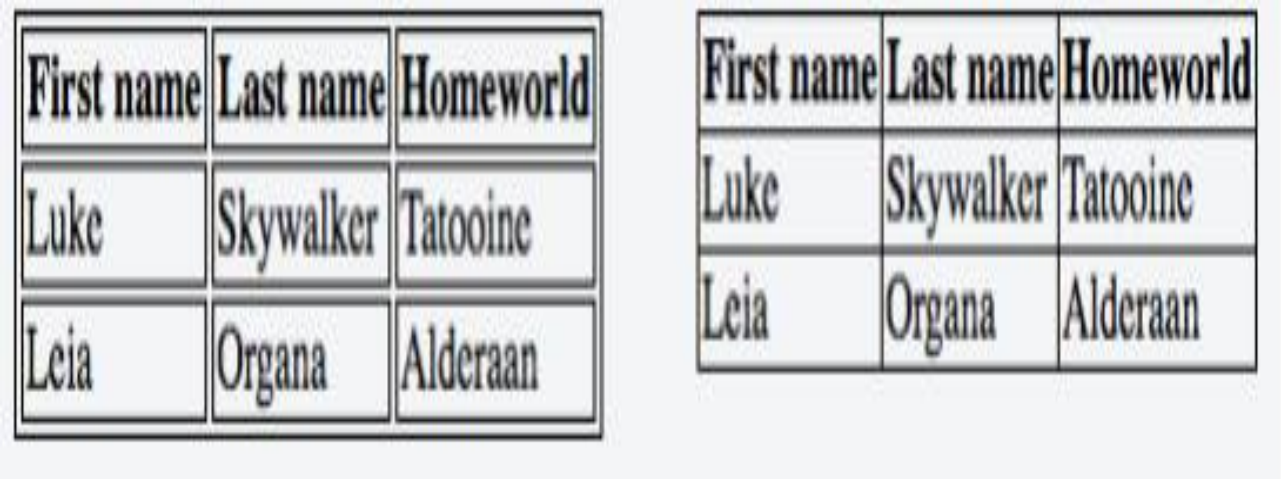
- The border-collapse property determines if a tables' borders should be separated or merged.
- Below an example of two tables with different values to the border-collapse property:

- The table on the left has border-collapse: separate while the one on the right has border-collapse: collapse.

Value Description

separate This is the default value. It makes the borders of the table separate from each other.

collapse This value sets the borders of the table to merge together, rather than being distinct.



First name	Last name	Homeworld
Luke	Skywalker	Tatooine
Leia	Organa	Alderaan

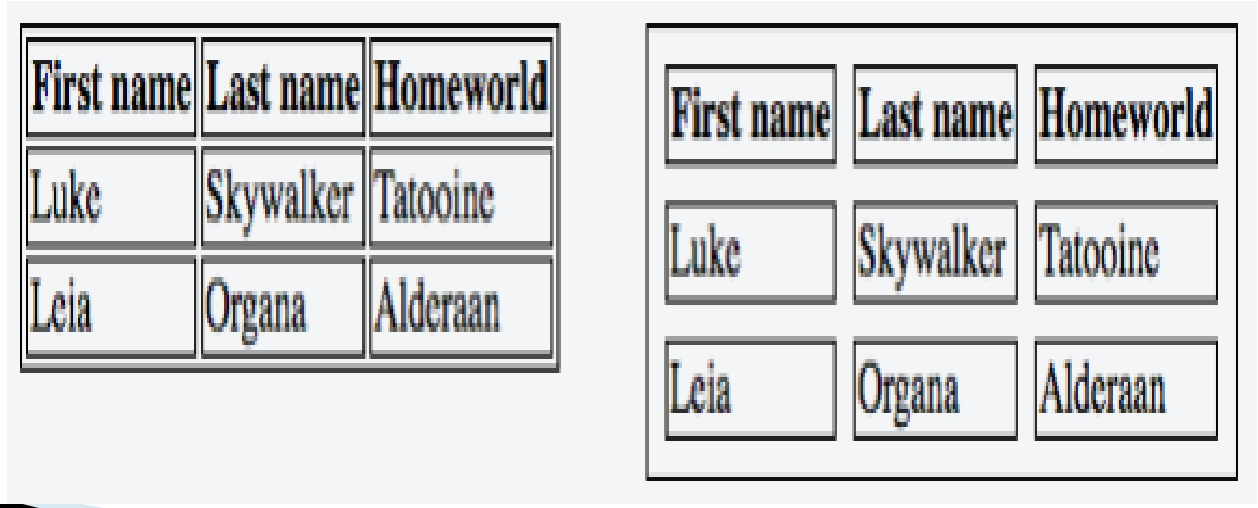
First name	Last name	Homeworld
Luke	Skywalker	Tatooine
Leia	Organa	Alderaan

border-spacing

- The border-spacing property determines the spacing between cells.
- This has no effect unless border-collapse is

set to separate.

- Below an example of two tables with different values to the border-spacing property:



First name	Last name	Homeworld
Luke	Skywalker	Tatooine
Leia	Organa	Alderaan

First name	Last name	Homeworld
Luke	Skywalker	Tatooine
Leia	Organa	Alderaan

- The table on the left has border-spacing: 2px (default) while the one on the right has border-spacing: 8px.

Value Description

<length> This is the default behavior, though the exact value can vary between browsers.

<length> <length> This syntax allows specifying separate horizontal and vertical values respectively.

caption-side:

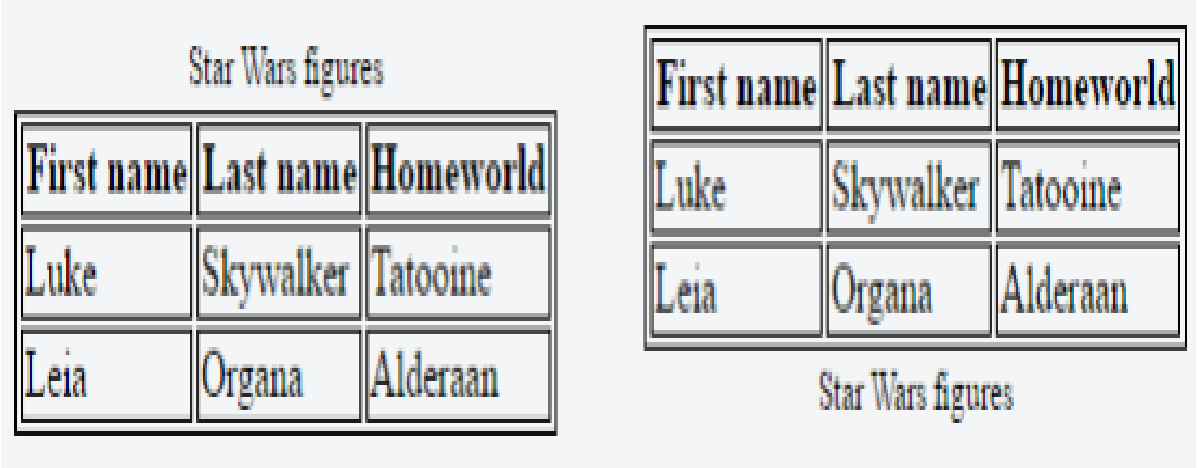
- The caption-side property determines the vertical positioning of the <caption> element within a table.
- This has no effect if such element does not exist.
- Below an example with two tables with different values set to the caption-side property:

- The table on the left has caption-side: top while the one on the right has caption-side: bottom.

Value Description

top This is the default value. It places the caption above the table.

bottom This value places the caption below the table.



Star Wars figures

First name	Last name	Homeworld
Luke	Skywalker	Tatooine
Leia	Organa	Alderaan

First name	Last name	Homeworld
Luke	Skywalker	Tatooine
Leia	Organa	Alderaan

Star Wars figures

Animations

Transition:

Parameter **Details**

property Either the CSS property to transition on, or all, which specifies all transition-able properties.

duration Transition time, either in seconds or milliseconds.

timing-function Specifies a function to define how intermediate values for properties are computed. Common values are ease, linear, and step-end. Check out the easing function cheatsheet for more.

delay Amount of time, in seconds or milliseconds, to wait before playing the animation.

@keyframes

[from | to | <percentage>] You can either specify a set time with a percentage value, or two percentage values, ie 10%, 20%, for a period of time where the keyframe's set attributes are set.

block Any amount of CSS attributes for the keyframe.

Animations with keyframes:

- For multi-stage CSS animations, you can create CSS @keyframes.
- Keyframes allow you to define multiple animation points, called a keyframe, to define more complex animations.

Basic Example

- In this example, we'll make a basic background animation that cycles between all colors.

@keyframes rainbow-background {

```
0% { background-color: #ff0000; }
 8.333% { background-color: #ff8000; }
 16.667% { background-color: #ffff00; }
 25.000% { background-color: #80ff00; }
 33.333% { background-color: #00ff00; }
 41.667% { background-color: #00ff80; }
 50.000% { background-color: #00ffff; }
 58.333% { background-color: #0080ff; }
 66.667% { background-color: #0000ff; }
 75.000% { background-color: #8000ff; }
 83.333% { background-color: #ff00ff; }
 91.667% { background-color: #ff0080; }
 100.00% { background-color: #ff0000; }
}
```

```
.RainbowBackground {
  animation: rainbow-background 5s infinite;
}
```

- The actual animation property takes the following arguments.

animation-name: The name of our animation. In this case, rainbow-background

animation-duration: How long the animation will take, in this case 5 seconds.

animation-iteration-count (Optional): The number of times the animation will loop. In this case, the animation will go on indefinitely. By default, the animation will play once.

animation-delay (Optional): Specifies how long to wait before the animation starts.

- It defaults to 0 seconds,

and can take negative values.

- For example, -2s would start the animation 2 seconds into its loop.

animation-timing-function (Optional): Specifies the speed curve of the animation.

- It defaults to ease, where the animation starts slow, gets faster and ends slow.
- In this particular example, both the 0% and 100% keyframes specify { background-color: #ff0000; }. Wherever two or more keyframes share a state, one may specify them in a single statement.

In this case, the two 0% and 100%

lines could be replaced with this single line:

0%, 100% { background-color: #ff0000; }

Cross-browser compatibility

- For older WebKit-based browsers, you'll need to use the vendor prefix on both the @keyframes declaration and theanimation property, like so:

@-webkit-keyframes{}

-webkit-animation: ...

2D Transforms:

Function/Parameter **Details**

rotate(x) Defines a transformation that moves the element around a fixed point on the Z axis

translate(x,y) Moves the position of the element on the X and Y axis

translateX(x) Moves the position of the element on the X axis

translateY(y) Moves the position of the element on the Y axis

scale(x,y) Modifies the size of the element on the X and Y axis

scaleX(x) Modifies the size of the element on the X axis

scaleY(y) Modifies the size of the element on the Y axis

skew(x,y) Shear mapping, or transvection, distorting each point of an element by a certain angle in each direction

skewX(x) Horizontal shear mapping distorting each point of an element by a certain angle in the horizontal direction

skewY(y) Vertical shear mapping distorting each point of an element by a certain angle in the vertical direction

Rotate:

HTML

```
<div class="rotate"></div>
```

CSS

```
.rotate {
 width: 100px;
 height: 100px;
 background: teal;
 transform: rotate(45deg);
}
```

- This example will rotate the div by 45 degrees clockwise. The center of rotation is in the center of the div, 50% from left and 50% from top.
- You can change the center of rotation by setting the transform-origin property.

transform-origin: 100% 50%;

- The above example will set the center of rotation to the middle of the right side end.

Scale

HTML

```
<div class="scale"></div>
```

CSS

```
.scale {
 width: 100px;
 height: 100px;
 background: teal;
 transform: scale(0.5, 1.3);
}
```

- This example will scale the div to 100px * 0.5 = 50px on the X axis and to 100px * 1.3 = 130px on the Y axis.

- The center of the transform is in the center of the div, 50% from left and 50% from top.

Skew

HTML

```
<div class="skew"></div>
```

CSS

```
.skew {
 width: 100px;
 height: 100px;
 background: teal;
 transform: skew(20deg, -30deg);
}
```

- This example will skew the div by 20 degrees on the X axis and by - 30 degrees on the Y axis.
- The center of the transform is in the center of the div, 50% from left and 50% from top.

Multiple transforms

- Multiple transforms can be applied to an element in one property like this:

transform: rotate(15deg) translateX(200px);

- This will rotate the element 15 degrees clockwise and then translate it 200px to the right.
- In chained transforms, the coordinate system moves with the element

Changing the order of the transforms will change the output.

The first example will be different to

transform: translateX(200px) rotate(15deg);

```
<div class="transform"></div>
.transform {
  transform: rotate(15deg) translateX(200px);
}
```

Transform Origin:

- Transformations are done with respect to a point which is defined by the transform-origin property.
- The property takes 2 values : transform-origin: X Y;
- In the following example the first div (.tl) is rotate around the top left corner with transform-origin: 0 0; and
- the second (.tr)is transformed around it's top right corner with transform-origin: 100% 0.
- The rotation is applied

on hover :

HTML:

```
<div class="transform originl"></div>
<div class="transform origin2"></div>
```

CSS:

```
.transform {
 display: inline-block;
 width: 200px;
 height: 100px;
 background: teal;
 transition: transform 1s;
}
.origin1 {
 transform-origin: 0 0;
}
.origin2 {
 transform-origin: 100% 0;
}
```

```
. transform:hover {
 transform: rotate(30deg);
}
```

The default value for the transform-origin property is 50% 50% which is the center of the element

3D Transforms

Compass pointer or needle shape using 3D transforms

CSS

```
div.needle {
 margin: 100px;
 height: 150px;
 width: 150px;
 transform: rotateY(85deg) rotateZ(45deg);
```

```
/* presentational */
 background-image: linear-gradient(to top left, #555 0%, #555 40%, #444 50%, #333 97%);
 box-shadow: inset 6px 6px 22px 8px #272727;
}
```

HTML

```
<div class='needle'></div>
```

- The output of the above example would be a needle resting on its tip.
- For creating a needle that is resting on its base, the rotation should be along the X-axis instead of along Y-axis.
- So the transform property's value would have to be something like rotateX(85deg) rotateZ(45deg);.

3D text effect with shadow

HTML:

```
<div id="title">
 <h1 data-content="HOVER">HOVER</h1>
</div>
```

CSS:

```
*{margin:0;padding:0;}
html,body{height:100%;width:100%;overflow:hidden;back
ground:#0099CC;}
#title{
 position:absolute;
 top:50%; left:50%;
 transform:translate(-50%,-50%);
```

```
 perspective-origin:50% 50%;
 perspective:300px;
}
h1{
 text-align:center;
 font-size:12vmin;
 font-family: 'Open Sans', sans-serif;
 color:rgba(0,0,0,0.8);
 line-height:1em;
 transform:rotateY(50deg);
 perspective:150px;
 perspective-origin:0% 50%;
}
h1:after{
 content:attr(data-content);
```

```
position:absolute;
 left:0;top:0;
 transform-origin:50% 100%;
 transform:rotateX(-90deg);
 color:#0099CC;
}
#title:before{
 content:'';
 position:absolute;
 top:-150%; left:-25%;
 width:180%; height:328%;
 background:rgba(255,255,255,0.7);
 transform-origin: 0 100%;
 transform: translatez(-200px) rotate(40deg)
skewX(35deg);
 border-radius:0 0 100% 0; }
```

Uplatz

backface-visibility

- The backface-visibility property relates to 3D transforms.
- With 3D transforms and the backface-visibility property, you're able to rotate an element such that the original front of an element no longer faces the screen.

```
<div class="flip">Loren ipsum</div>
<div class="flip back">Lorem ipsum</div>
.flip {
 -webkit-transform: rotateY(180deg);
 -moz-transform: rotateY(180deg);
 -ms-transform: rotateY(180deg);
 -webkit-backface-visibility: visible;
 -moz-backface-visibility: visible;
```

```
-ms-backface-visibility: visible;
}
.flip.back {
 -webkit-backface-visibility: hidden;
 -moz-backface-visibility: hidden;
 -ms-backface-visibility: hidden;
}
```

It has 4 values:

1. **visible (default) -** the element will always be visible even when not facing the screen.
2. **hidden** - the element is not visible when not facing the screen.
3. **inherit** - the property will gets its value from the its parent element
4. **initial -** sets the property to its default, which is visible

3D cube:

- 3D transforms can be use to create many 3D shapes. Here is a simple 3D CSS cube example:

HTML:

```
<div class="cube">
 <div class="cubeFace"></div>
 <div class="cubeFace face2"></div>
</div>
```

CSS:

```
body {
 perspective-origin: 50% 100%;
 perspective: 1500px;
 overflow: hidden;
}
.cube {
```

```
 position: relative;
 padding-bottom: 20%;
 transform-style: preserve-3d;
 transform-origin: 50% 100%;
 transform: rotateY(45deg) rotateX(0);
}
.cubeFace {
position: absolute;
top: 0;
left: 40%;
width: 20%;
height: 100%;
margin: 0 auto;
transform-style: inherit;
```

```
 background: #C52329;
 box-shadow: inset 0 0 0 5px #333;
 transform-origin: 50% 50%;
 transform: rotateX(90deg);
 backface-visibility: hidden;
}
.face2 {
 transform-origin: 50% 50%;
 transform: rotatez(90deg) translateX(100%)
rotateY(90deg);
}
.cubeFace:before, .cubeFace:after {
 content: '';
 position: absolute;
```

```
width: 100%;
 height: 100%;
 transform-origin: 0 0;
 background: inherit;
 box-shadow: inherit;
 backface-visibility: inherit;
}
.cubeFace:before {
 top: 100%;
 left: 0;
 transform: rotateX(-90deg);
}.cubeFace:after {
 top: 0;
 left: 100%;
 transform: rotateY(90deg);
}
```

Filter Property:

Value Description

blur(x) Blurs the image by x pixels.

brightness(x) Brightens the image at any value above 1.0 or 100%. Below that, the image will be darkened.

contrast(x) Provides more contrast to the image at any value above 1.0 or 100%. Below that, the image will get less saturated.

drop-shadow(h, v, x, y, z) Gives the image a drop-shadow. h and v can have negative values. x, y, and z are optional.

greyscale(x) Shows the image in greyscale, with a maximum value of 1.0 or 100%.

hue-rotate(x) Applies a hue-rotation to the image.

invert(x) Inverts the color of the image with a maximum value of 1.0 or 100%.

opacity(x) Sets how opaque/transparent the image is with a maximum value of 1.0 or 100%.

saturate(x) Saturates the image at any value above 1.0 or 100%. Below that, the image will start to de-saturate.

sepia(x) Converts the image to sepia with a maximum value of 1.0 or 100%

Blur

HTML

```
<img src='donald-duck.png' alt='Donald Duck' title='Donald Duck' />
```

CSS

```
img {
 -webkit-filter: blur(1px);
 filter: blur(1px);
}
```

Drop Shadow (use box-shadow instead if possible)

HTML

```
<p>My shadow always follows me.</p>
```

CSS

```
p {
 -webkit-filter: drop-shadow(10px 10px 1px green);
 filter: drop-shadow(10px 10px 1px green);
}
```

Hue Rotate:

HTML:

```
<img src='donaldduck.png' alt='Donald Duck'
title='Donald Duck' />
```

CSS:

```
img {
 -webkit-filter: hue-rotate(120deg);
 filter: hue-rotate(120deg);
}
```

Multiple Filter Values

- To use multiple filters, separate each value with a space.

HTML

```
<img src='donald-duck.png' alt='Donald Duck'
title='Donald Duck' />
```

CSS

```
img {
 -webkit-filter: brightness(200%) grayscale(100%) sepia(100%) invert(100%);
 filter: brightness(200%) grayscale(100%) sepia(100%) invert(100%);
}
```

box-shadow

Parameters Details

- inset by default, the shadow is treated as a drop shadow.
- the inset keyword draws the shadow inside the frame/border.

offset-x the horizontal distance

offset-y the vertical distance

)

- blur-radius 0 by default. value cannot be negative. the bigger the value, the bigger and lighter the shadow becomes.
- spread-radius 0 by default. positive values will cause the shadow to expand. negative values will cause the shadow to shrink.
- color can be of various notations: a color keyword, hexadecimal, rgb(), rgba(), hsl(), hsla()

bottom-only drop shadow using a pseudoelement:

HTML

```
<div class="box_shadow"></div>
```

CSS

```
.box_shadow {
 background-color: #1C90F3;
```

```
width: 200px;
 height: 100px;
 margin: 50px;
}
.box_shadow:after {
 content: "";
 width: 190px;
 height: 1px;
 margin-top: 98px;
 margin-left: 5px;
 display: block;
 position: absolute;
 z-index: -1;
 -webkit-box-shadow: 0px 0px 8px 2px #444444;
 -moz-box-shadow: 0px 0px 8px 2px #444444;
 box-shadow: 0px 0px 8px 2px #444444; }
```

inner drop shadow:

HTML:

```
<div class="box_shadow"></div>
```

CSS:

```
.box_shadow {
 background-color: #1C90F3;
 width: 200px;
 height: 100px;
 margin: 50px;
 -webkit-box-shadow: inset 0px 0px 10px 0px #444444;
 -moz-box-shadow: inset 0px 0px 10px 0px #444444;
 box-shadow: inset 0px 0px 10px 0px #444444;
}
```

multiple shadows:

HTML:

```
<div class="box_shadow"></div>
```

CSS:

```
.box_shadow {
 width: 100px;
 height: 100px;
 margin: 100px;
 box-shadow:
 -52px -52px 0px 0px #f65314,
 52px -52px 0px 0px #7cbb00,
 -52px 52px 0px 0px #00a1f1,
 52px 52px 0px 0px #ffbb00;
}
```

Summary:

- Grid
- Tables
- Transitions & Animation
- 2D Transforms & 3D Transforms
- Filter Property
- Box-shadow

Thank You……

If you have any quries please write to info@uplatz.com".