CSS3

BACKGROUNDS

With CSS3, we have a range of new properties and values to give us **more control over our background images**.

However, before we talk about these new properties and values, we need to understand **three special boxes**.

Three special boxes

content-box

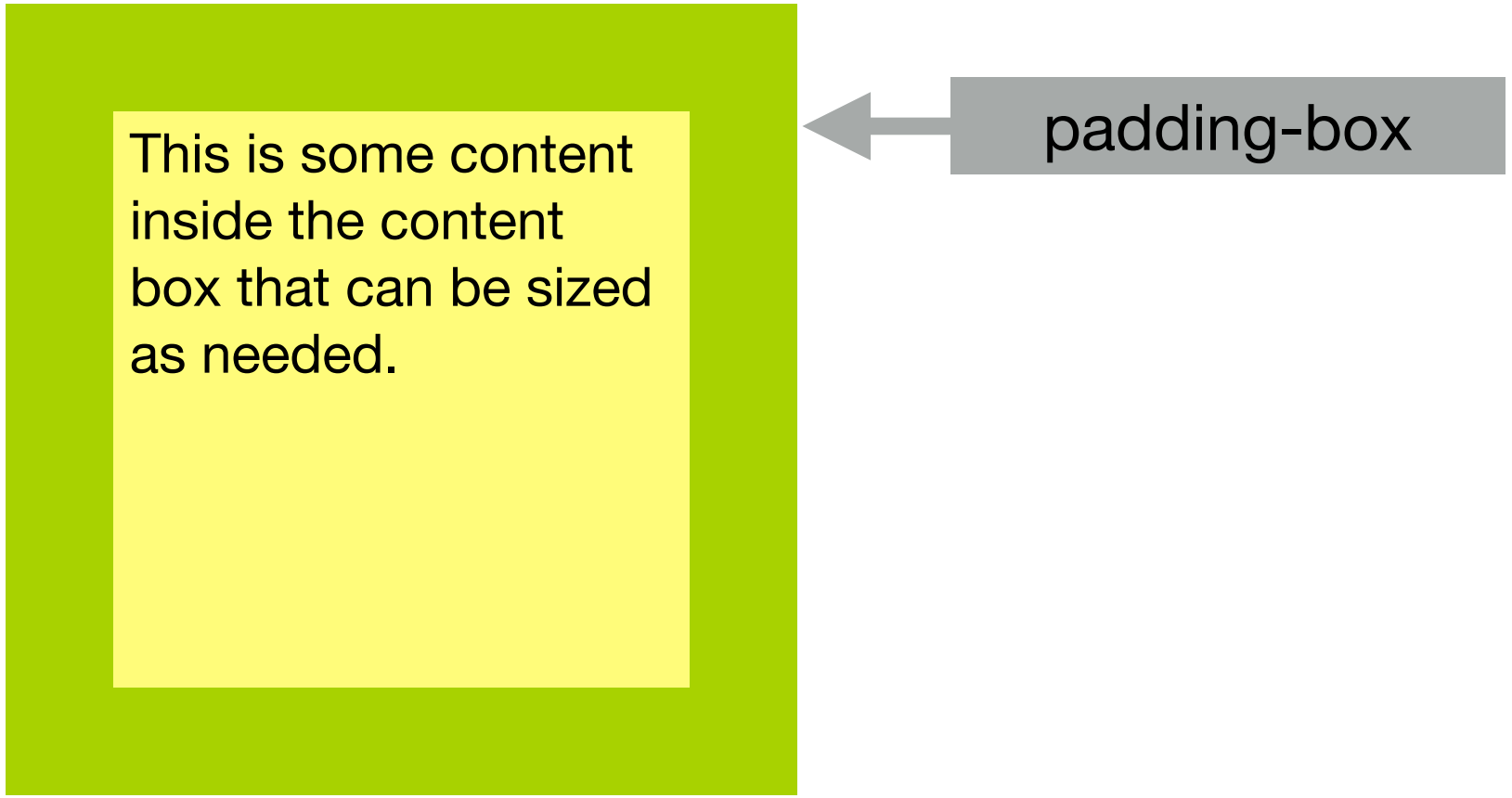
Let’s start with a simple container with some content inside. Although we cannot see it, there is an invisible box around the content called the **content-box**.

This is some content inside the content box that can be sized as needed.

content-box

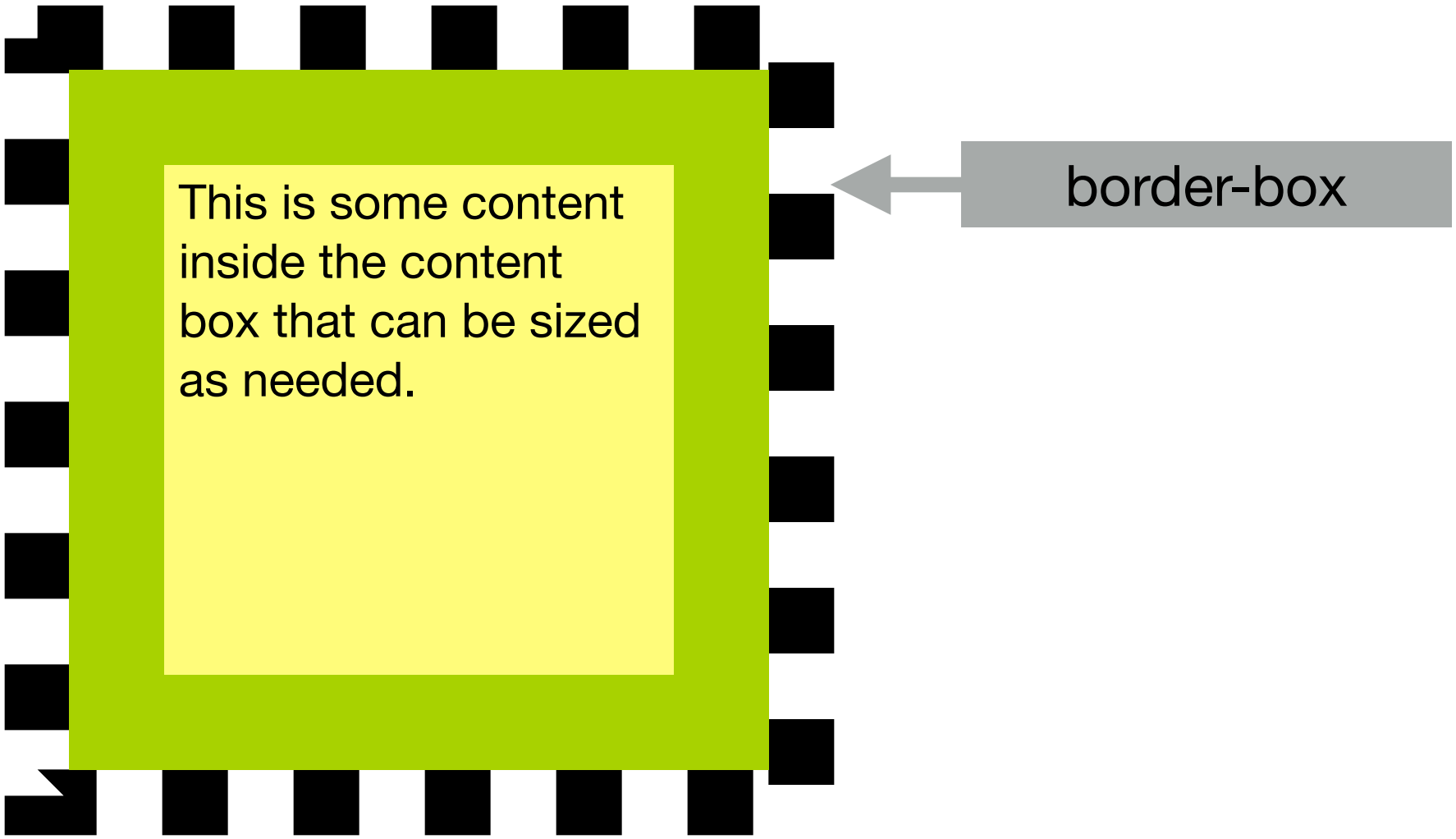
padding-box

If we were to add padding to all sides of this element, we would then have our second box, called a **padding-box**.


This is some content inside the content box that can be sized as needed.
padding-box

border-box

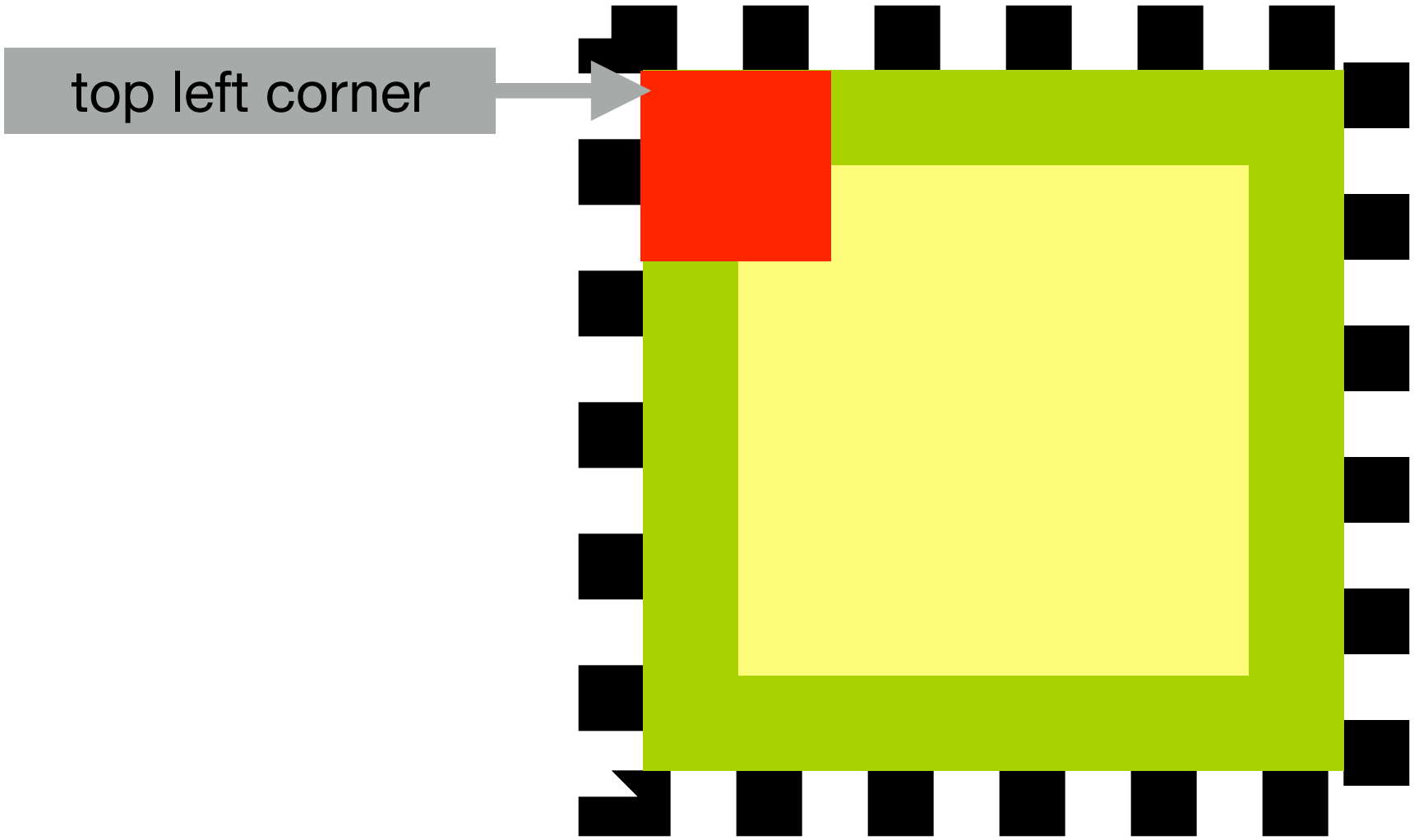
If we add a border around the padded-box, we would then have the third box, called a **border-box**.


This is some content inside the content box that can be sized as needed.
border-box

These boxes are **used to define** where background images are initially placed into boxes by browsers, how we can reposition these background images, and even how we can crop these background images.

background-position with four values

By default, background images are placed in the top left corner of the **padding-box**.


top left corner

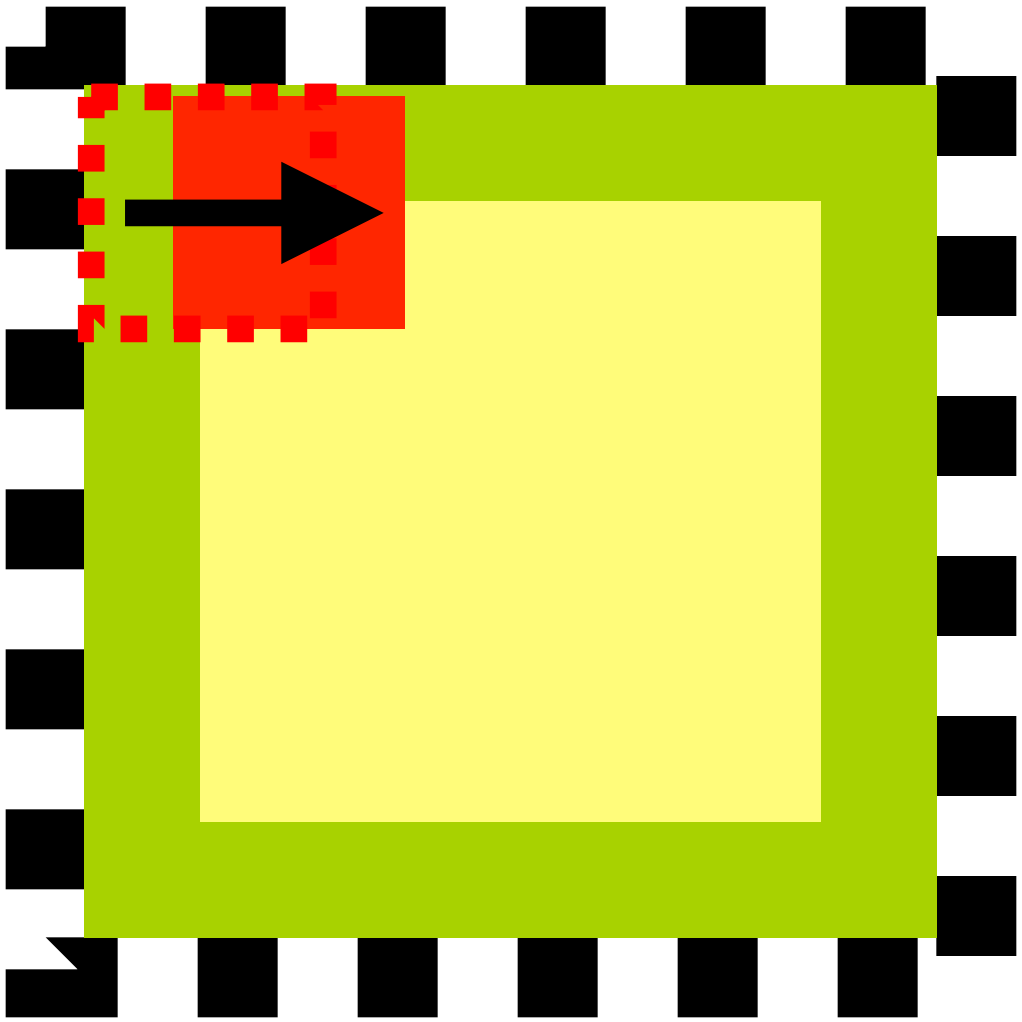
We can change this default position (or offset the position) by using the **background-position property**.

```
p { background-position: 5px 9px; }
```

In CSS2.1, we can use **two values** to determine the position of the background image in relation to the element.

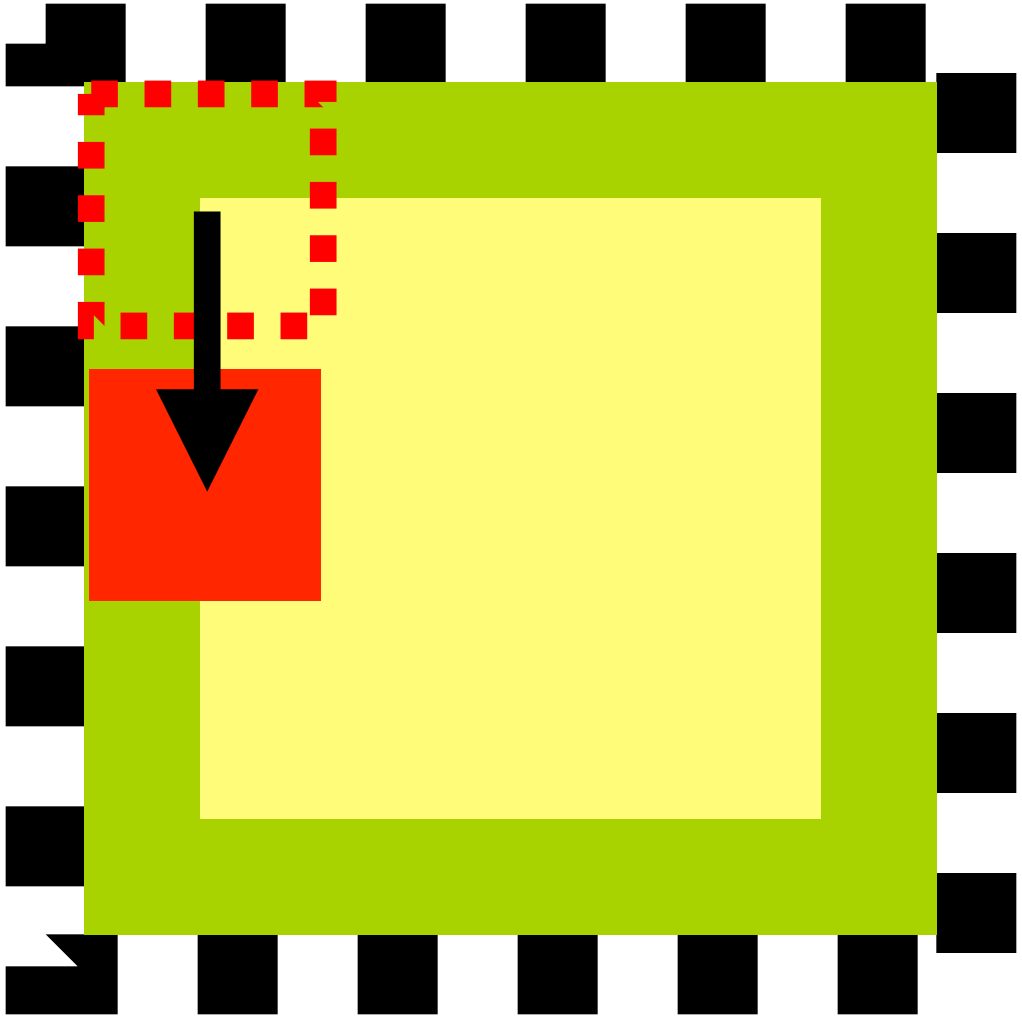
The first value represents the **horizontal position** (left or right) of the background-image.

```
p { background-position: 5px 20px; }
```

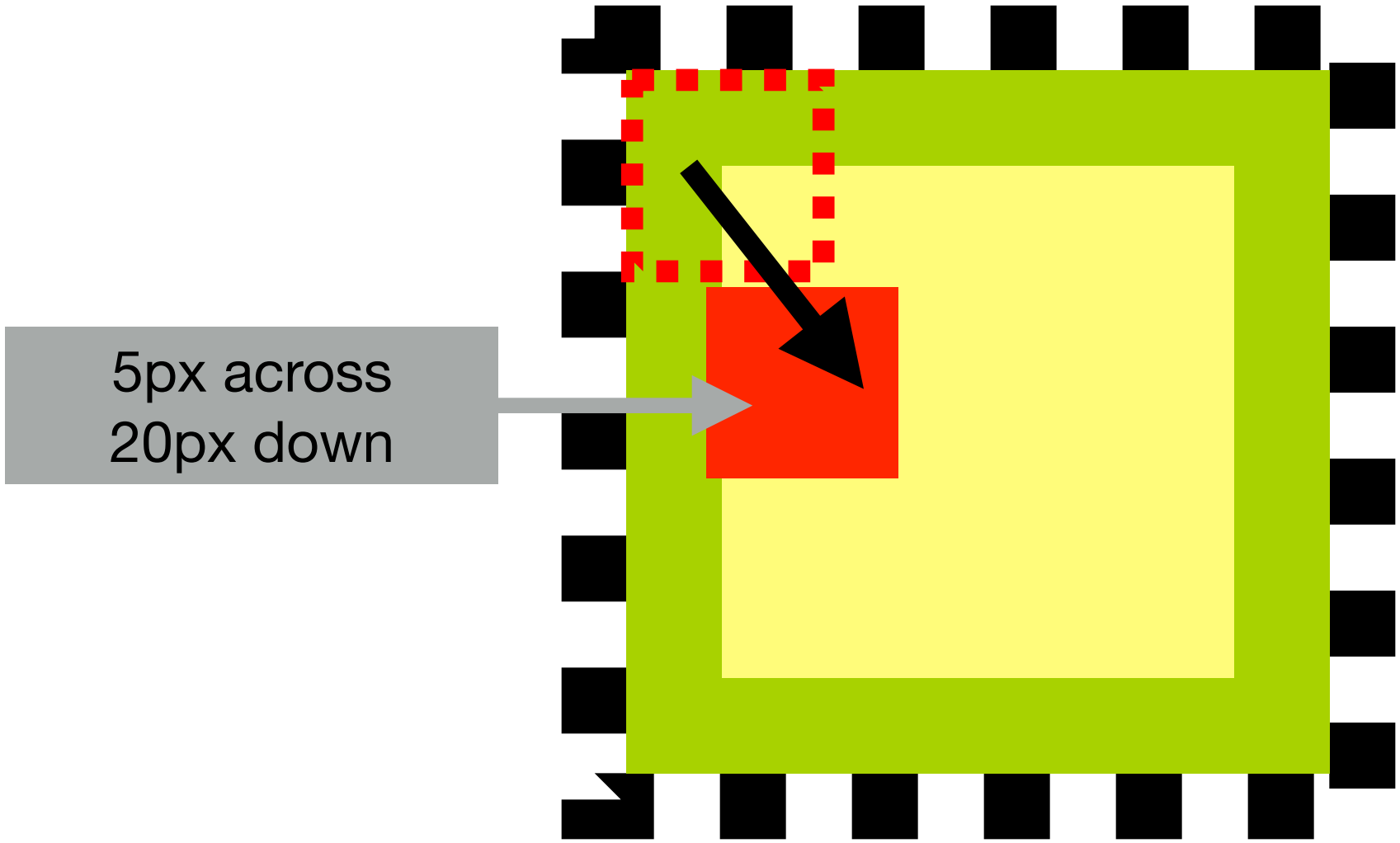


The second value represents the **vertical axis** (up or down) of the background-image.

```
p { background-position: 5px 20px; }
```



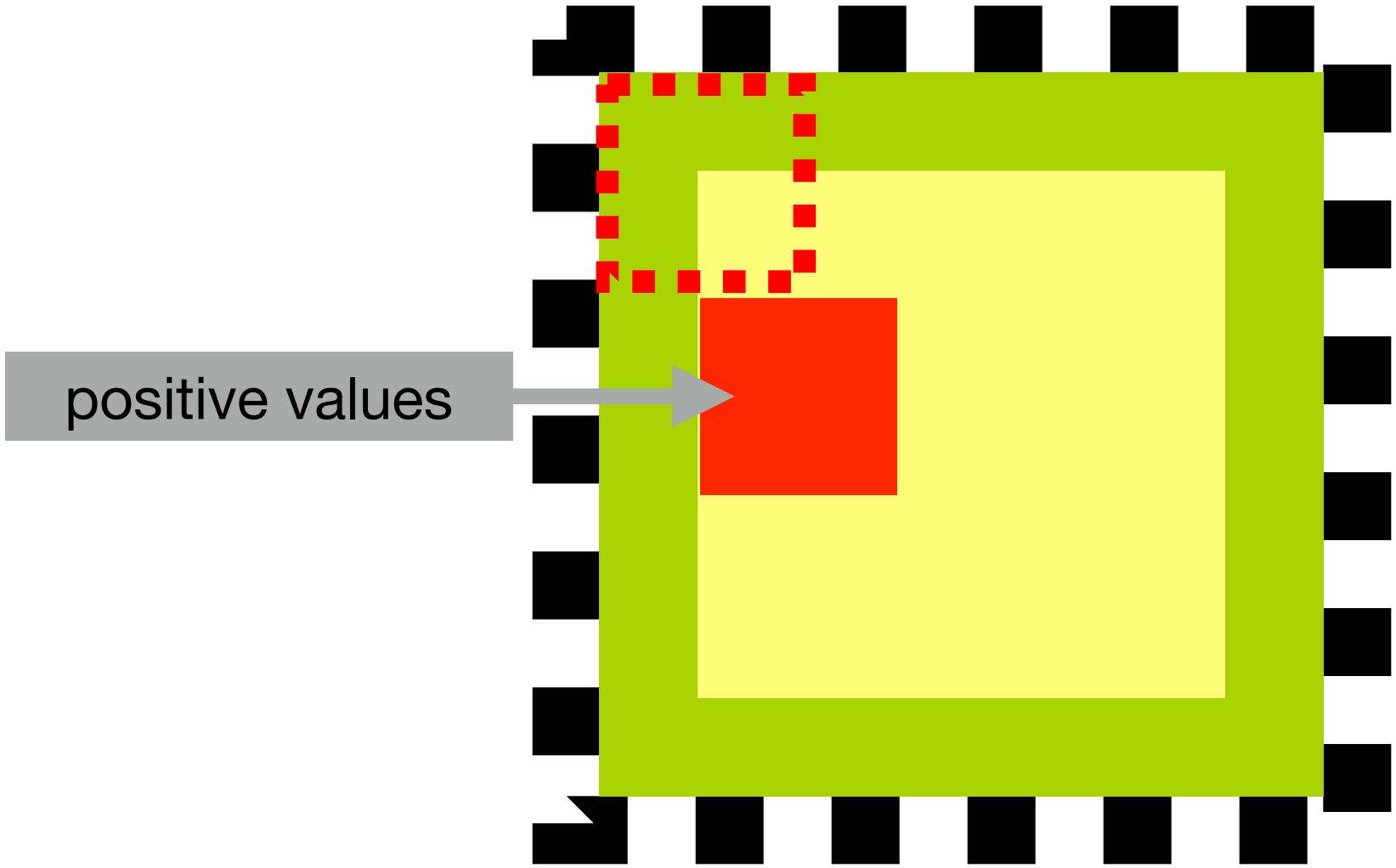
In this example, the background -image would be positioned 5px to the left, and 20px down **from the top left corner of the padding-box**.


5px across
20px down

We can use both **positive or negative values** to determine the position of background images.

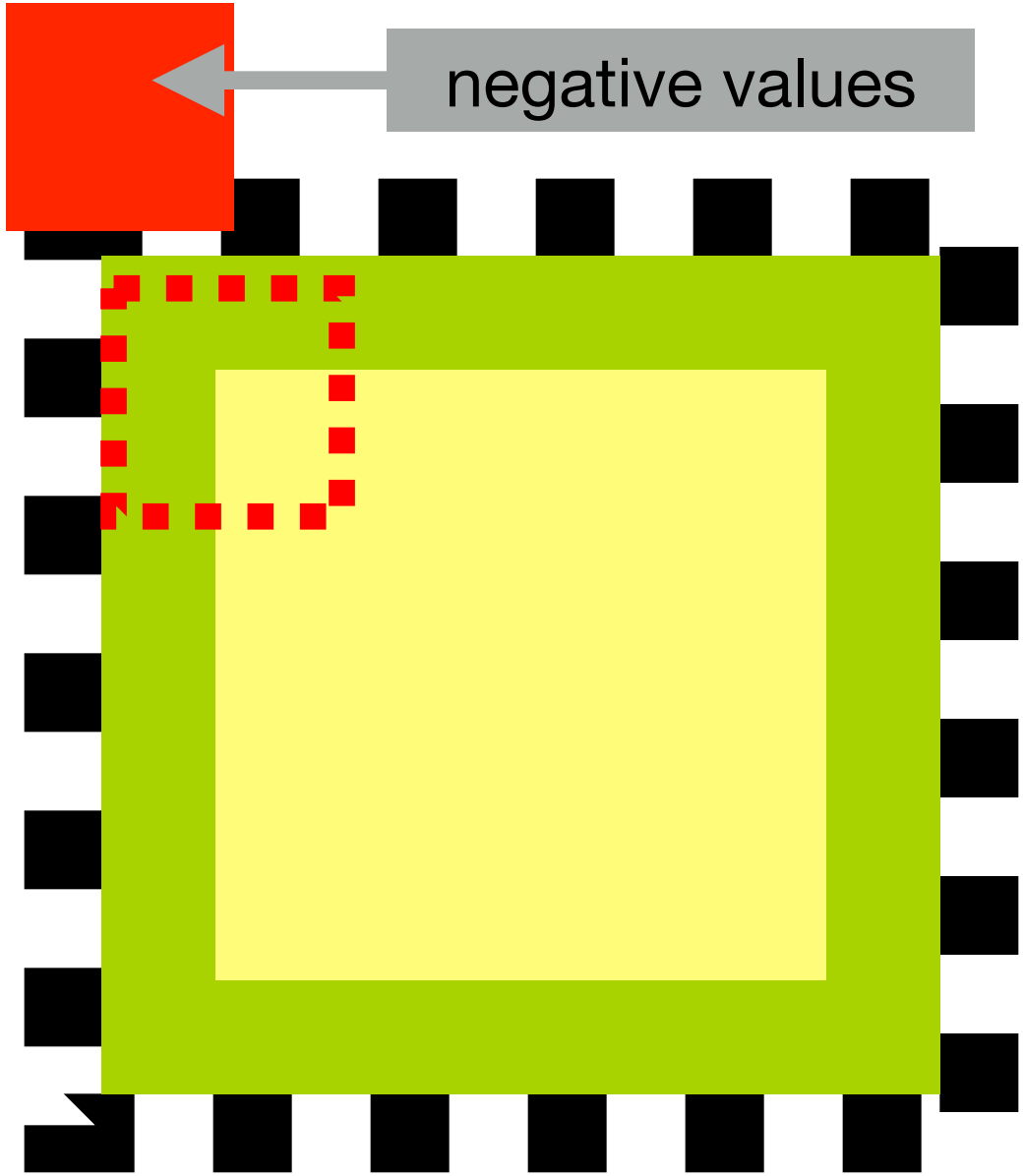
Positive values will move the background image to the **right and down** – inside the background area of the element.

```
p { background-position: 5px 20px; }
```


positive values

Negative values will move the background image to the **left and up** – out of the background area of the element.

```
p { background-position: -5px -20px; }
```

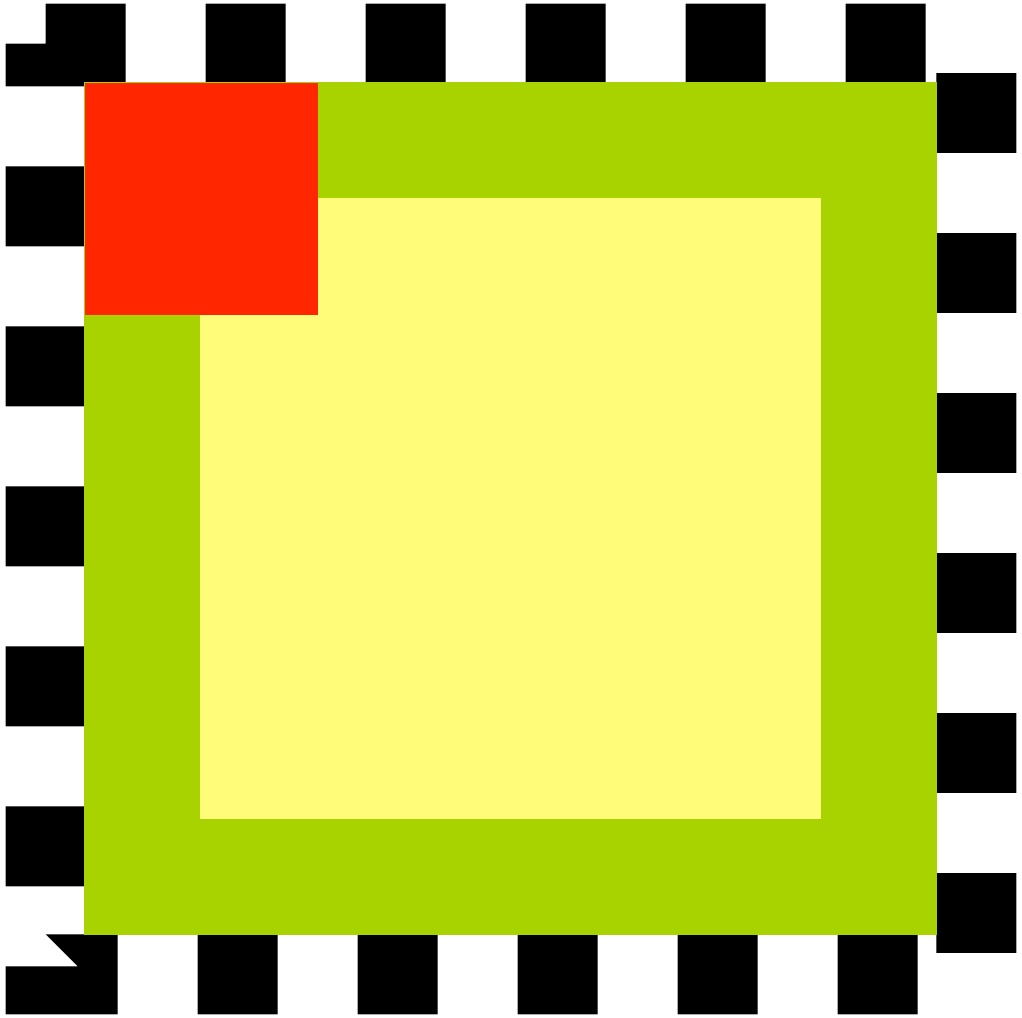

negative values

We can use **three different types of values** to define the horizontal and vertical axis. These are length values, percentage values or keywords

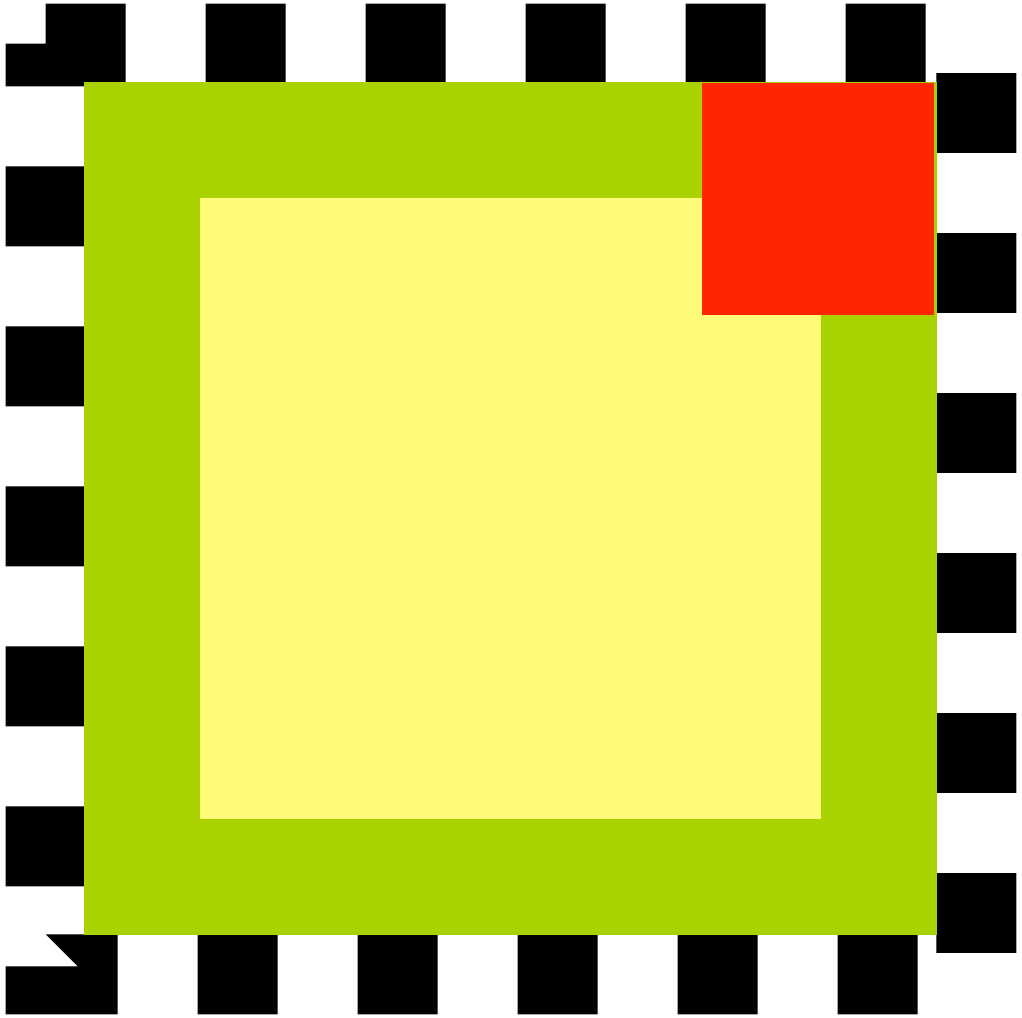
```
/* length values */
p { background-position: 10px 10px; }

/* percentage values */
p { background-position: 20% 50%; }

/* keyword values */
p { background-position: left bottom; }
```

<percentage> and <length> values represent an offset of the **top left corner** of the background image from the top left corner of the padding-box.

```
p { background-position: 10px 10px; }
p { background-position: 20% 50%; }
```



Keyword values position the background-image **relative to the value**. For example, the right value will place the right edge of the background-image against the right edge of the padding-box.



Keyword values include: left, centre right, top, centre and, bottom.

```
p { background-position: left top; }
p { background-position: center center; }
p { background-position: right bottom; }
```

We could use **any combination** of these three types of values to position a background image.

```
p { background-position: 10px 50%; }
p { background-position: 20% bottom; }
p { background-position: left 10px; }
```

If only **one value is defined**, the second value is assumed to be “center”.

```
p { background-position: 5px; }

p { background-position: 5px [center]; }
```

In CSS3, we can specify up to **four values** for background-position. The first two values represent the horizontal axis. The second two values represent the vertical axis.

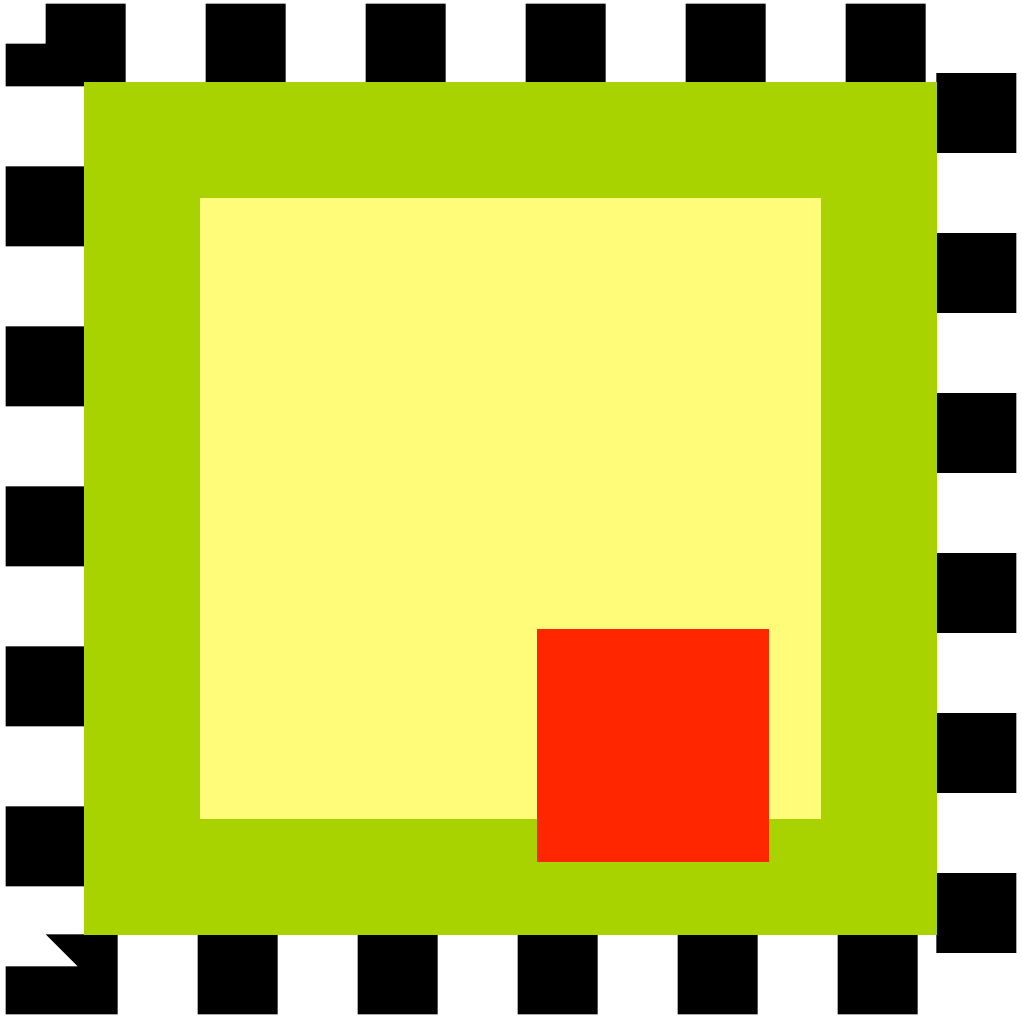
```
p { background-position:
    left 10px top 15px; }
```

If three or four values are given, then each <percentage> or<length> represents an offset and **must be preceded by a keyword**, which specifies from which edge the offset is given.

```
p { background-position:
    left 10px
    top 15px; }
```

For example, **“right 20px bottom 10px”** represents a “20px” horizontal offset to the left from the right edge and a “10px” vertical offset up from the bottom edge

```
p { background-position:
    right 20px bottom 10px; }
```

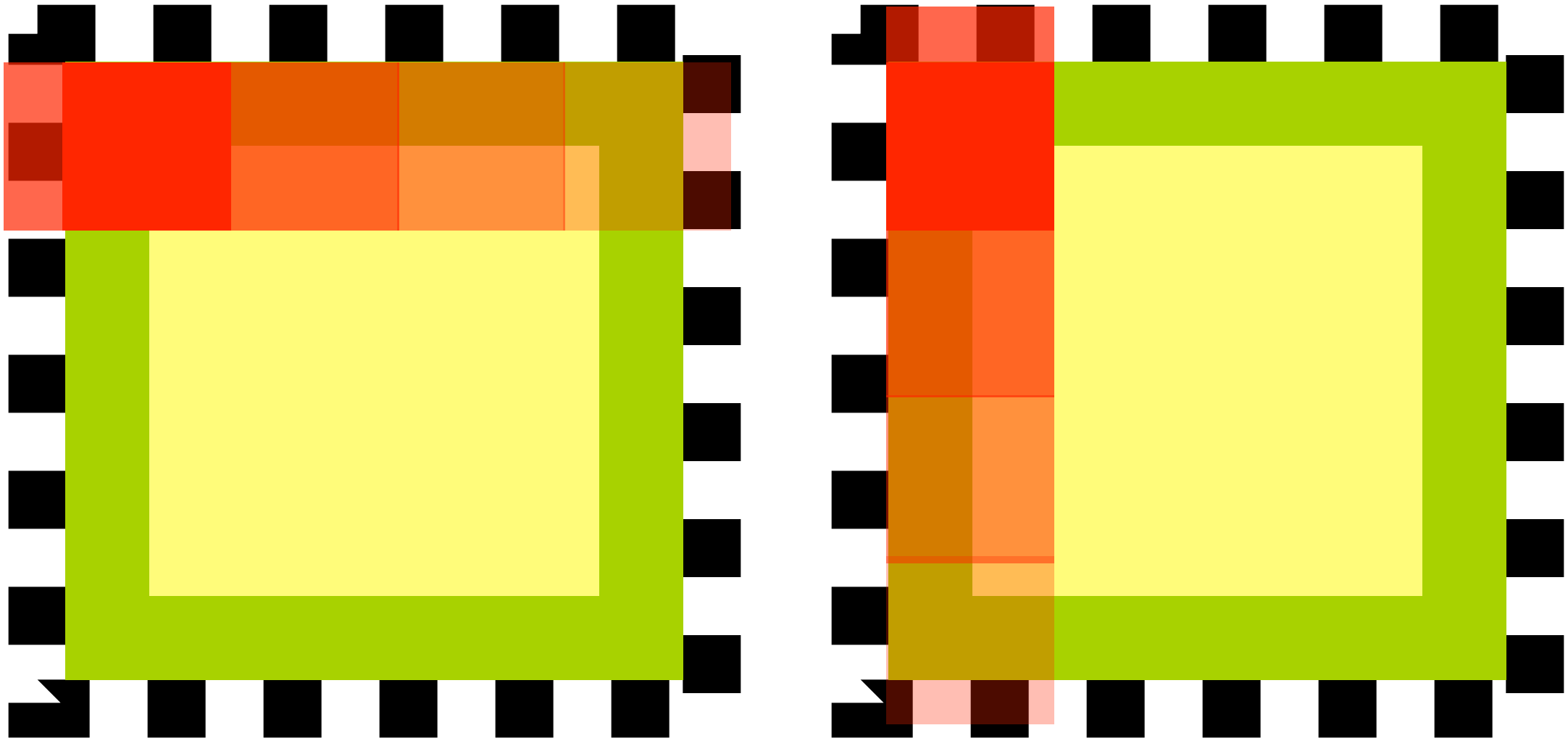


This is a powerful addition, as it means we can position images using length values or percentage values **in relation to any of the four corners of elements**, not just the top left corner.

background-repeat with new values

By default, images will repeat along both the x and y axis – starting from the top left corner of the padding-box.

Even though the background images start in the top left corner of the padding-box, they will repeat outwards in all directions, including into the border-box area.



In CSS2.1, we could change the repeat behaviour using **four different keywords**.

```
p { background-repeat: repeat; }
p { background-repeat: repeat-x; }
p { background-repeat: repeat-y; }
p { background-repeat: no-repeat; }
```

In CSS3, we can now define background-repeat using **two values** instead of one.

The first of these two values represents the **horizontal axis**. The second value represents the **vertical axis**.

```
p { background-repeat: repeat no-repeat; }
```

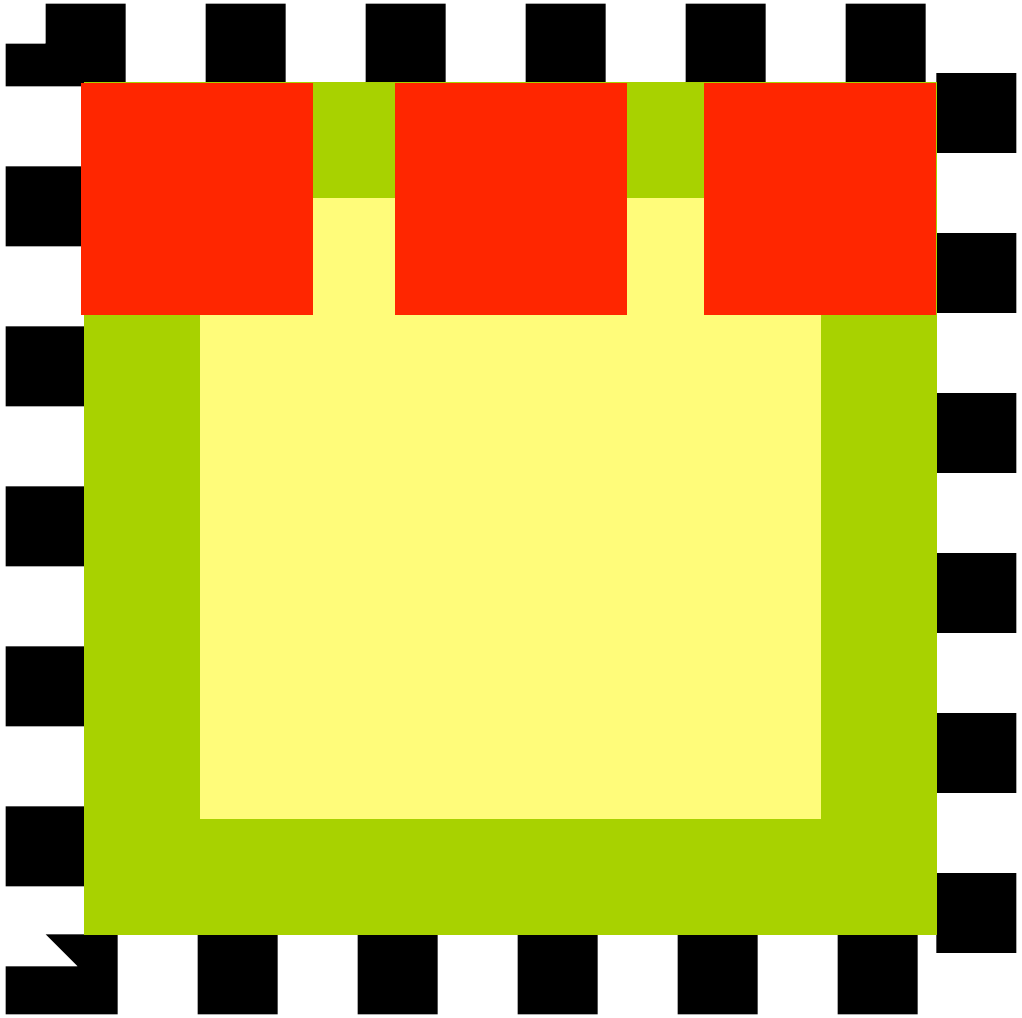
If we use one value only, the browsers will interpret this as a **double value**. This allows the background-repeat value to be backwards compatible.

```
p { background-repeat: repeat [repeat]; }
```

CSS3 also allows us to use two new values with the background-repeat property – they are **space** and **round**.

The **space value** sets the image to repeat as often as will fit within the background area and then the images are spaced out to fill the area. The first and last images touch the edges of the area.

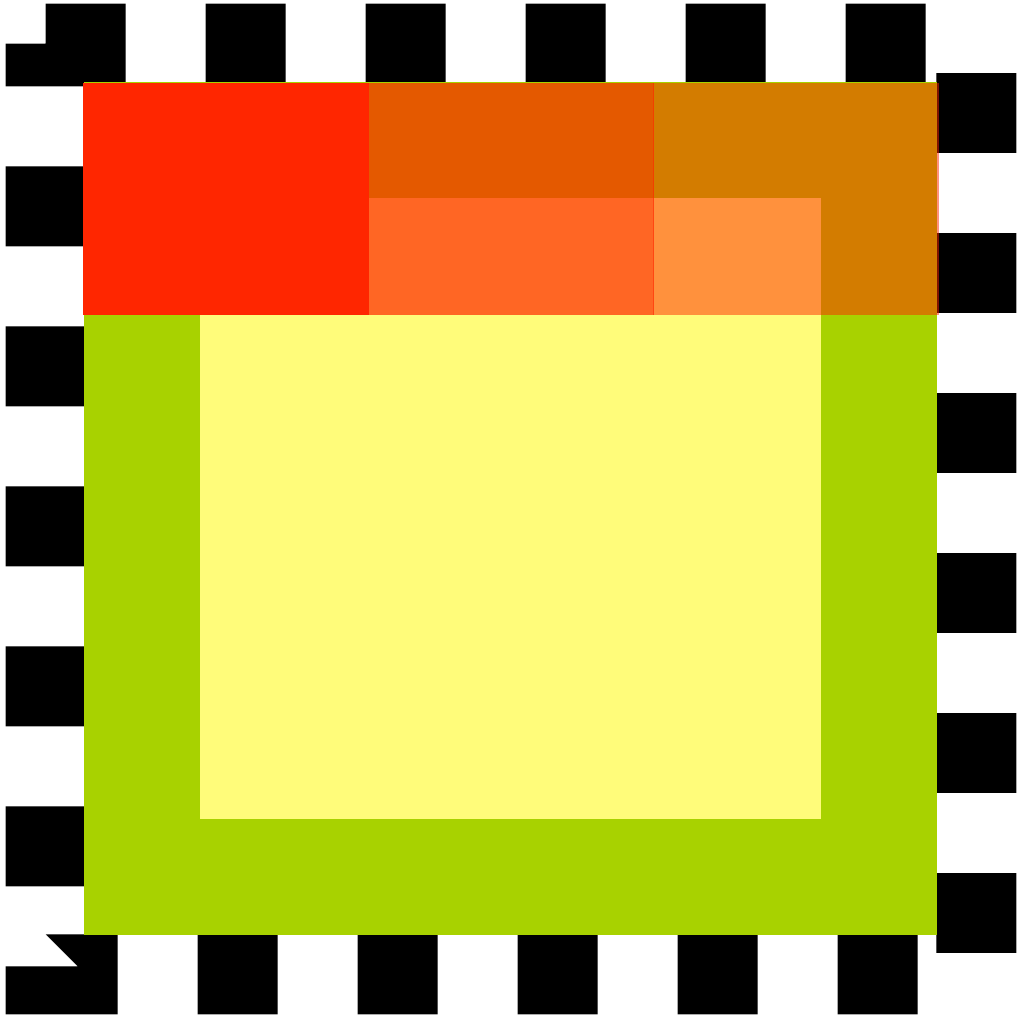
```
p { background-repeat: space; }
```



The **round value** sets the image to repeat as often as will fit within the background area. If it doesn’t fit a whole number of times, it is rescaled so that it will fit into the container’s dimensions.

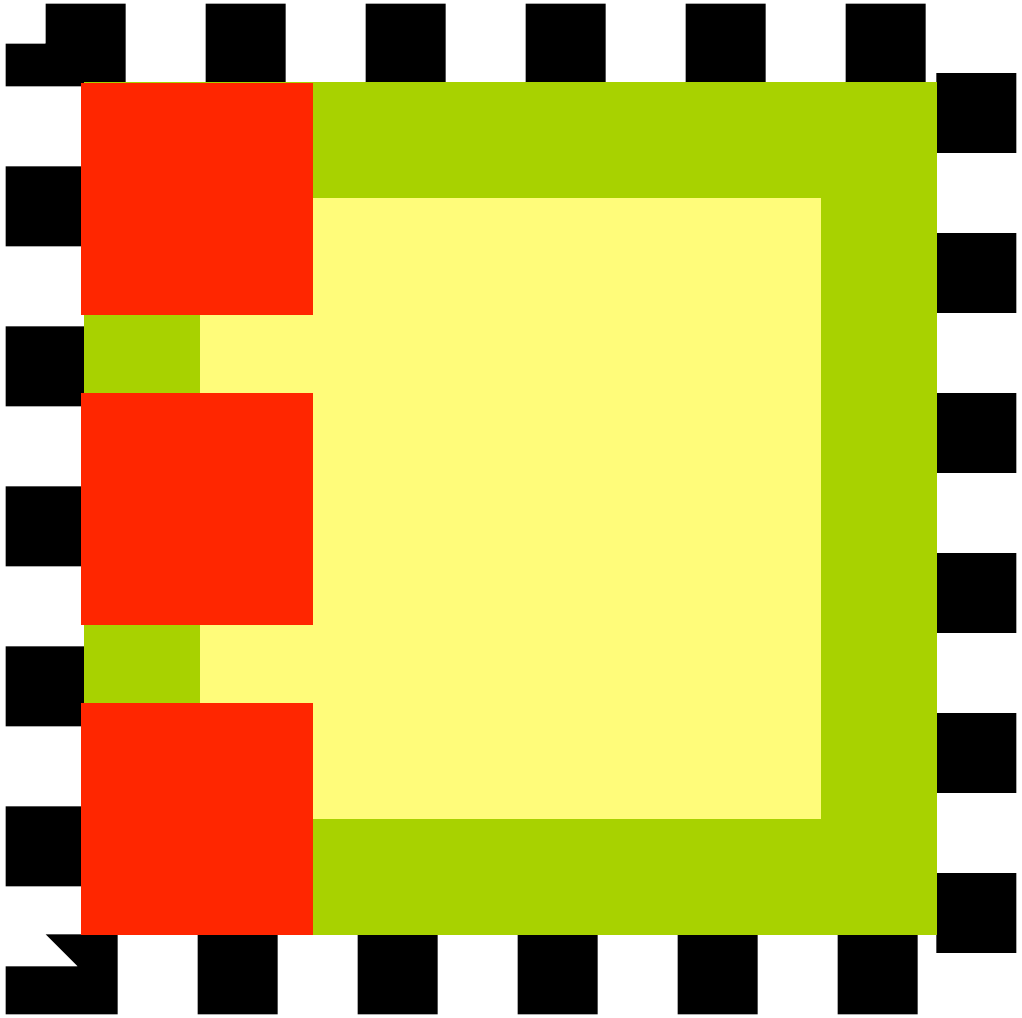
Be aware that the background images may be **stretched or distorted** using this method.

```
p { background-repeat: round; }
```



These new values give us **much more flexibility** when laying out background-images. For example, we can now use two values to define different horizontal and vertical behaviour.

```
p { background-repeat: no-repeat space; }
```



New background properties

CSS2.1 has **five background properties** that we can use to control the background of elements.

```
p
{
    background-color: XXX;
    background-image: XXX;
    background-repeat: XXX;
    background-attachment: XXX;
    background-position: XXX;
}
```

In CSS3, we can also use **three new background properties**.

```
p
{
    background-color: XXX;
    background-image: XXX;
    background-repeat: XXX;
    background-attachment: XXX;
    background-position: XXX;
    background-origin: XXX;
    background-clip: XXX;
    background-size: XXX;
}
```

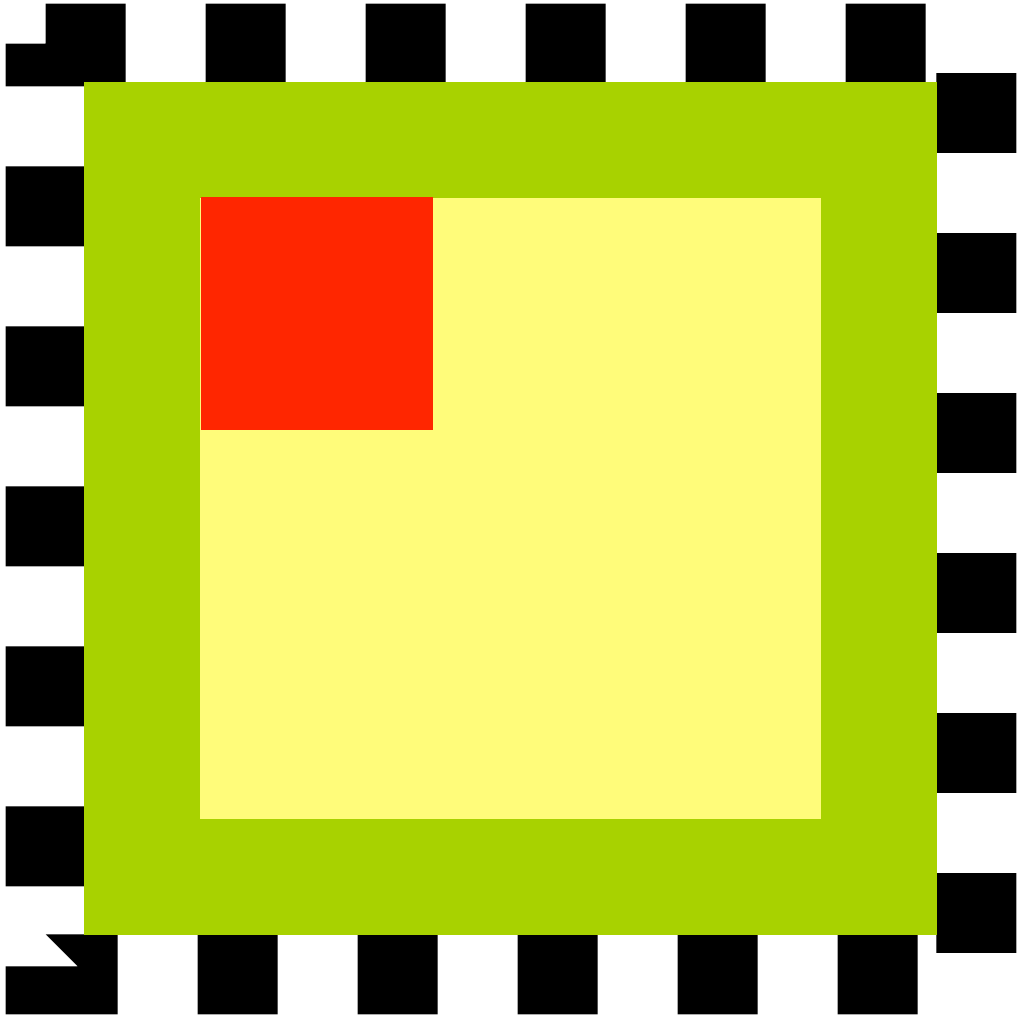
background-origin

The **background-origin property** is used to determine where background images are positioned inside a box.

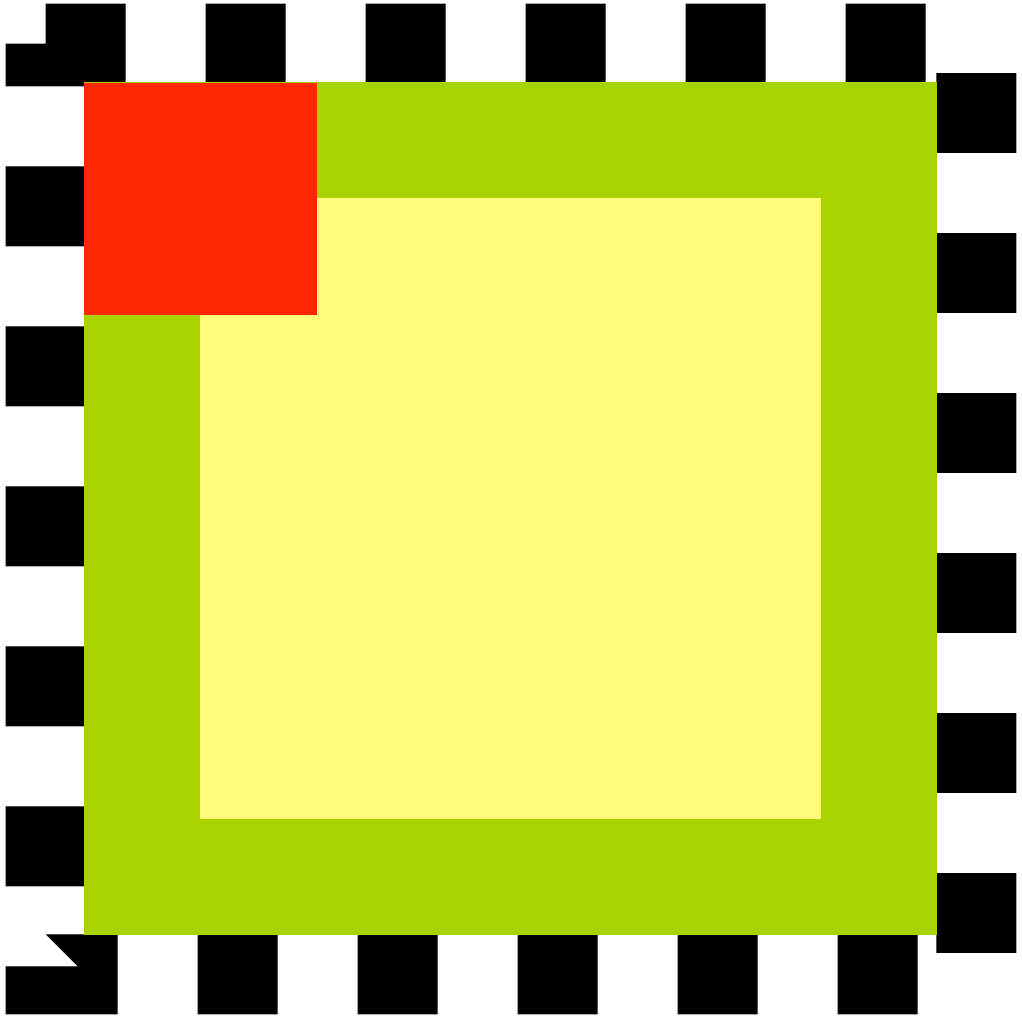
```
div { background-origin: padding-box; }
```

We can position our background images using **one of three** background-origin values:

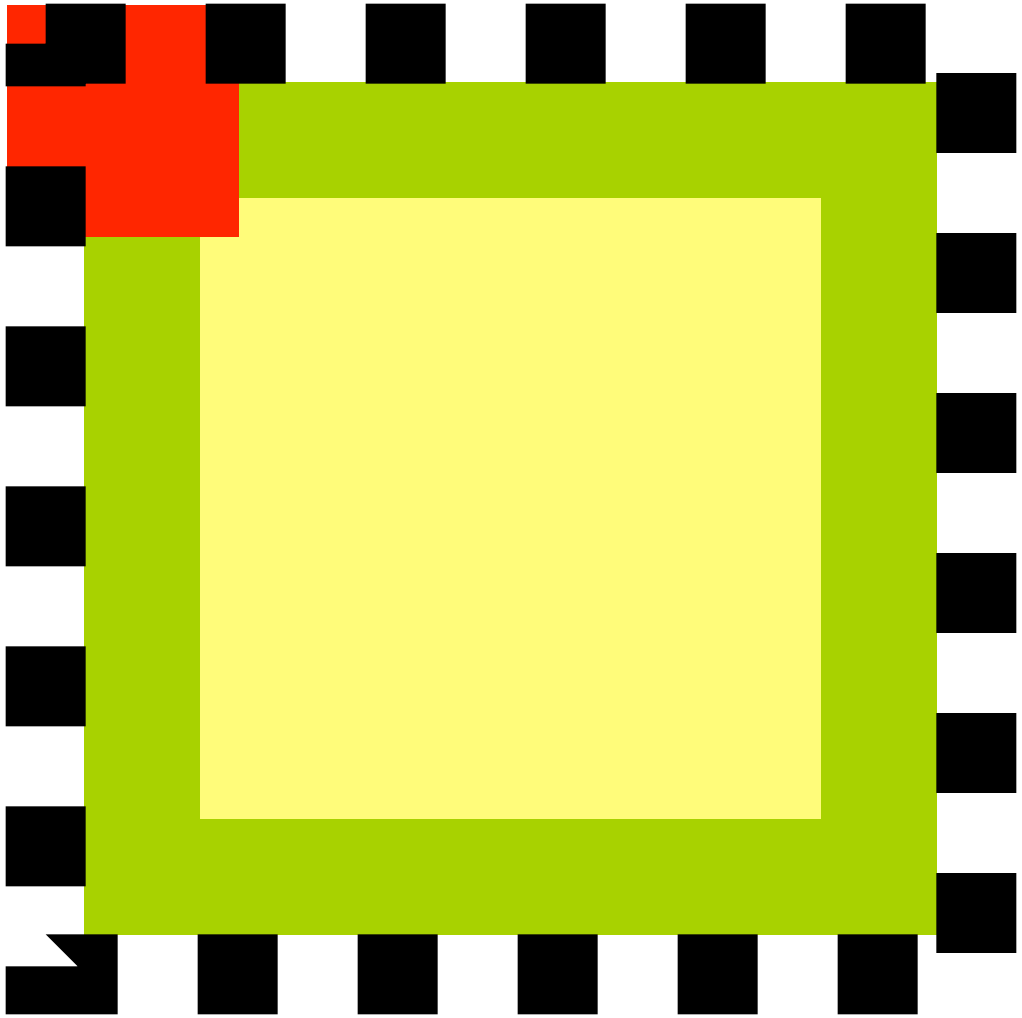
```
div { background-origin: content-box; }
```



```
div { background-origin: padding-box; }
```



```
div { background-origin: border-box; }
```



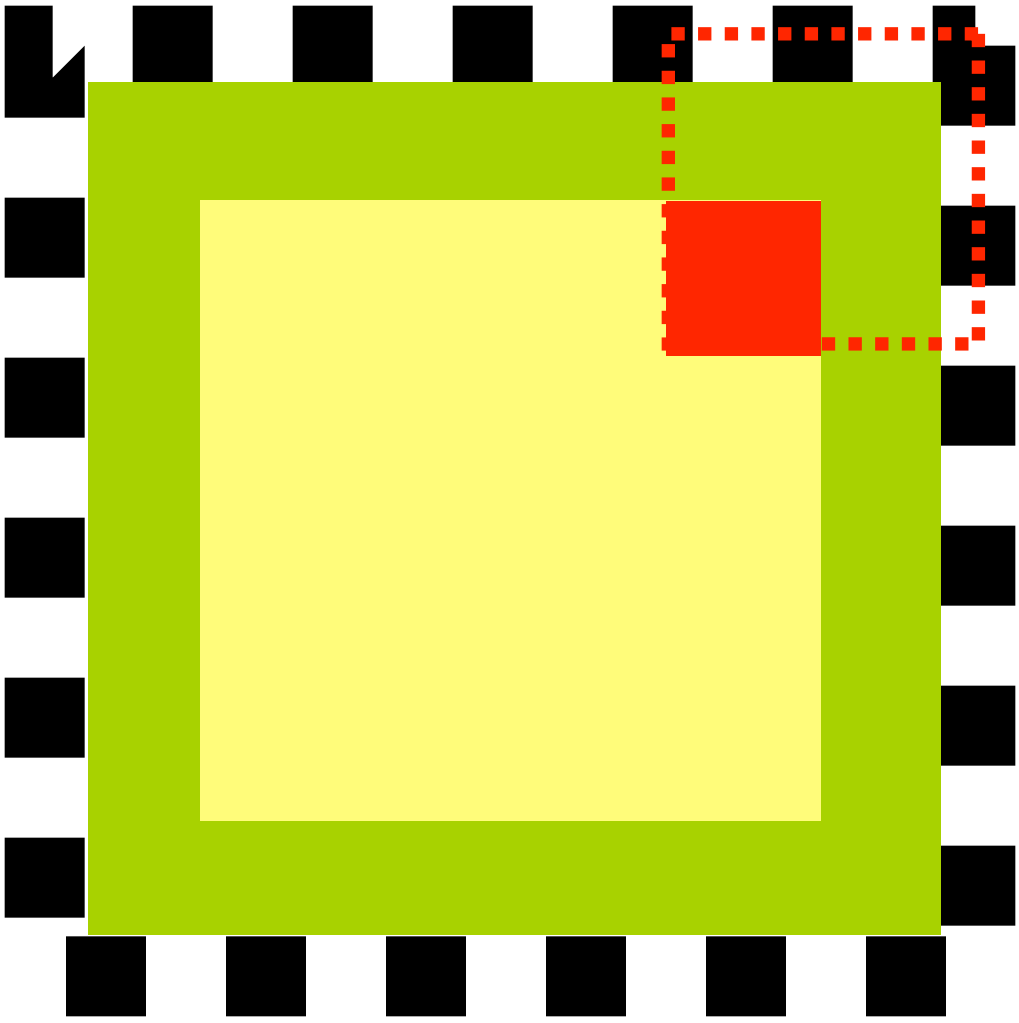
background-clip

The **background-clip property** is used to determine where and if background images are clipped (or cut off) inside the background area.

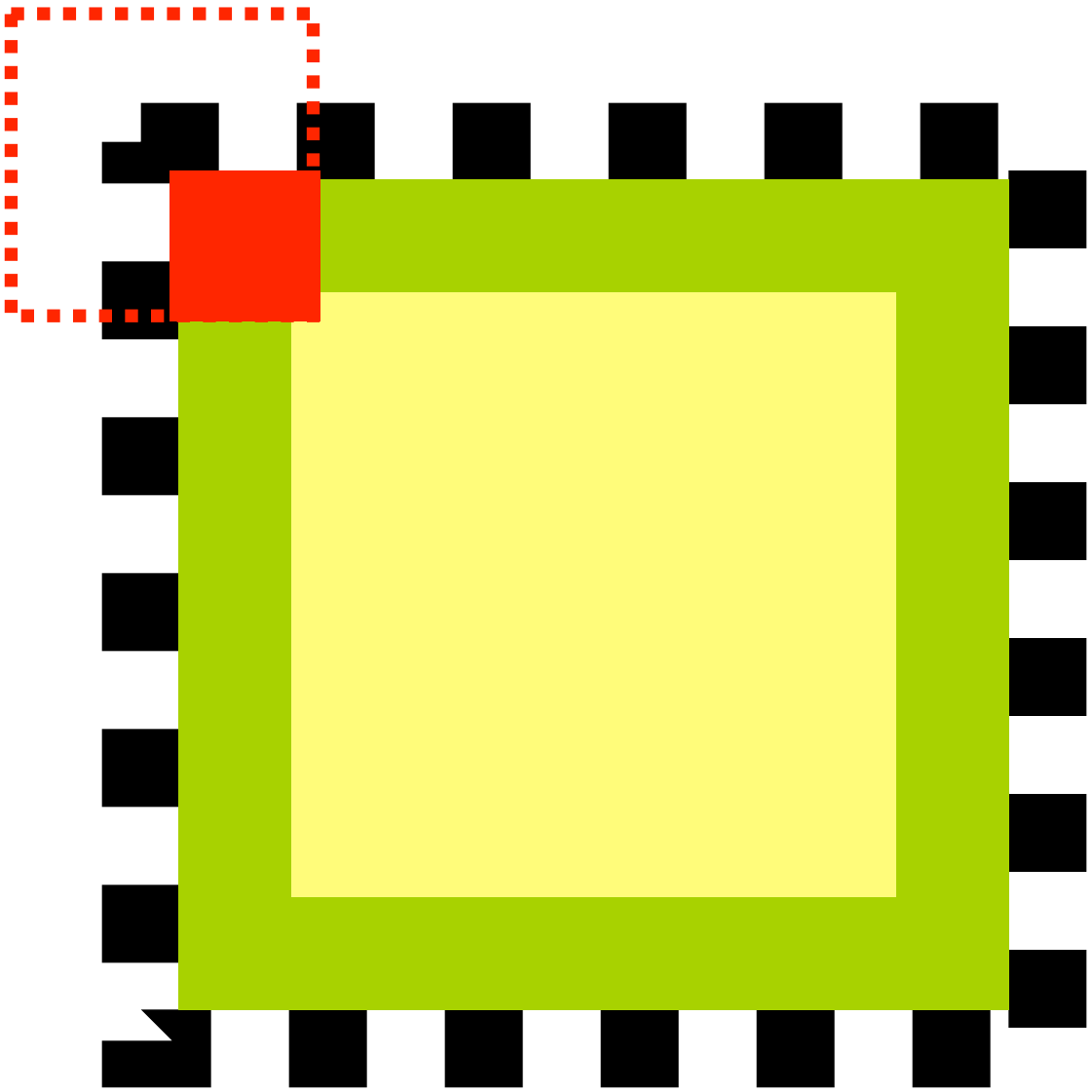
```
div { background-clip: padding-box; }
```

We can clip our background images via the background-clip property using **three possible values**:

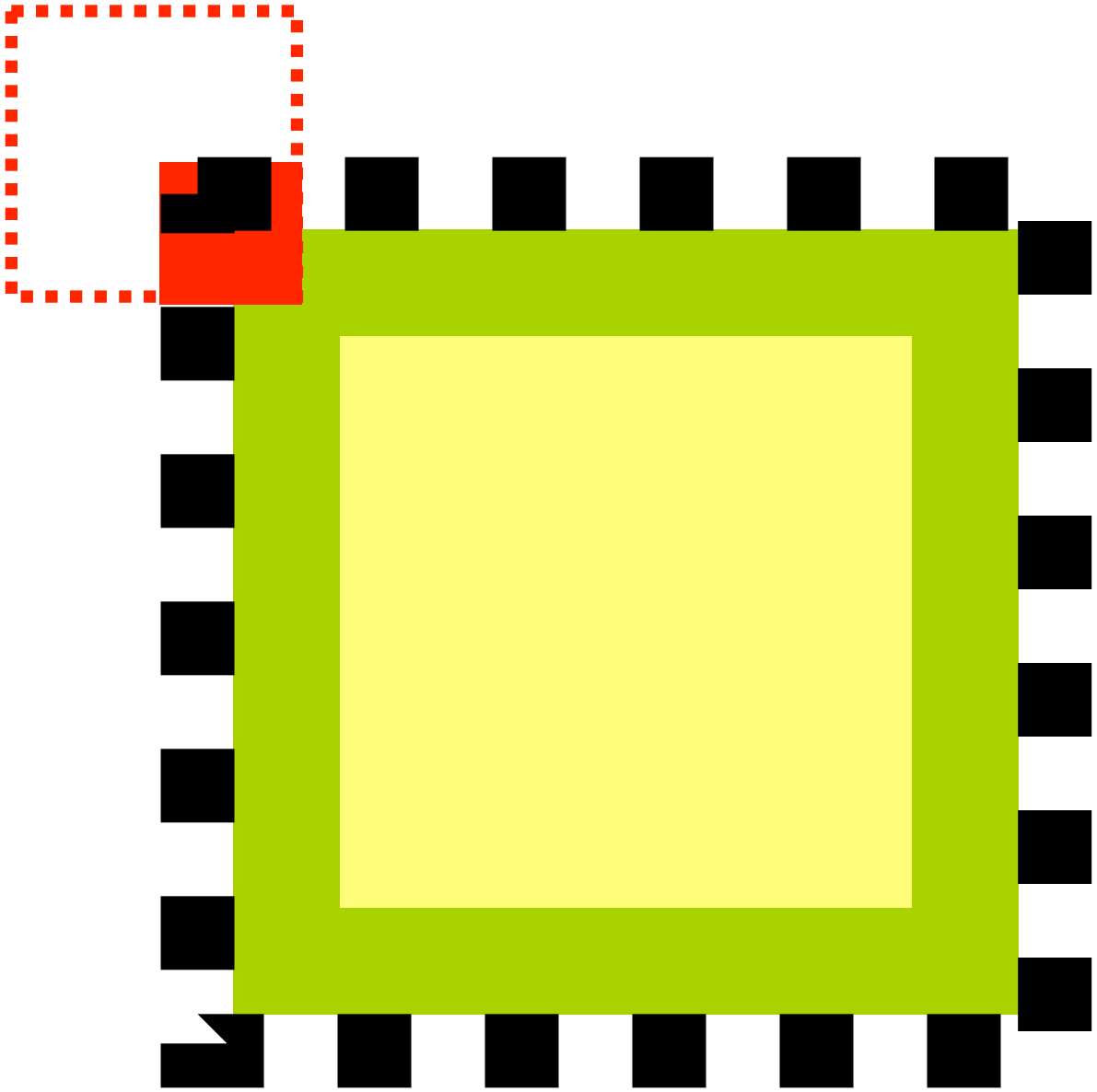
```
div { background-clip: content-box; }
```



```
div { background-clip: padding-box; }
```



```
div { background-clip: border-box; }
```



background-size

In CSS2.1, we could apply background images to elements, but we had **no way to control the size** of these background images.

However, CSS3 allows us to set the size of our background images using the **background-size property**.

```
div { background-size: 10px 20px; }
```

We can set the background-size using **three different types of values**:

length values
percentage values
keyword values

length values

The length value sets the **height and width** of the background image. The first value sets the width, the second value sets the height.

```
div { background-size: 10px 20px; }
```

If only one length value is given, the second value is set to the ‘initial value’ of **auto**.

```
div { background-size: 10px; }

div { background-size: 10px [auto]; }
```

percentage values

The **percentage value** sets the height and width to a percent of the parent element. The first value sets the width, the second value sets the height.

```
div { background-size: 20% 40%; }
```

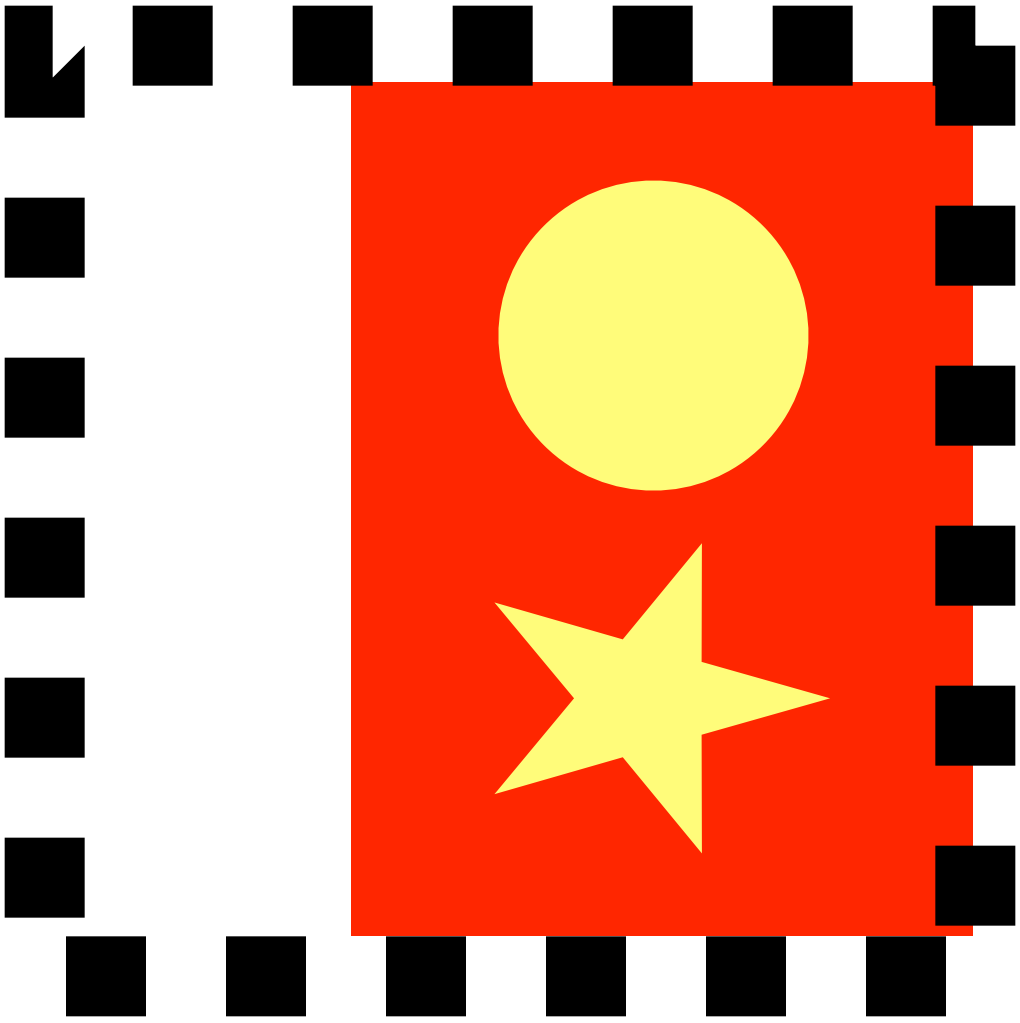
If only one percentage value is given, the second is set to the ‘initial value’ of **auto**.

```
div { background-size: 20%; }

div { background-size: 20% [auto]; }
```

contain keyword value

The **contain keyword value** will scale the image (while keeping its aspect ratio) so that the entire image can fit inside the background area.

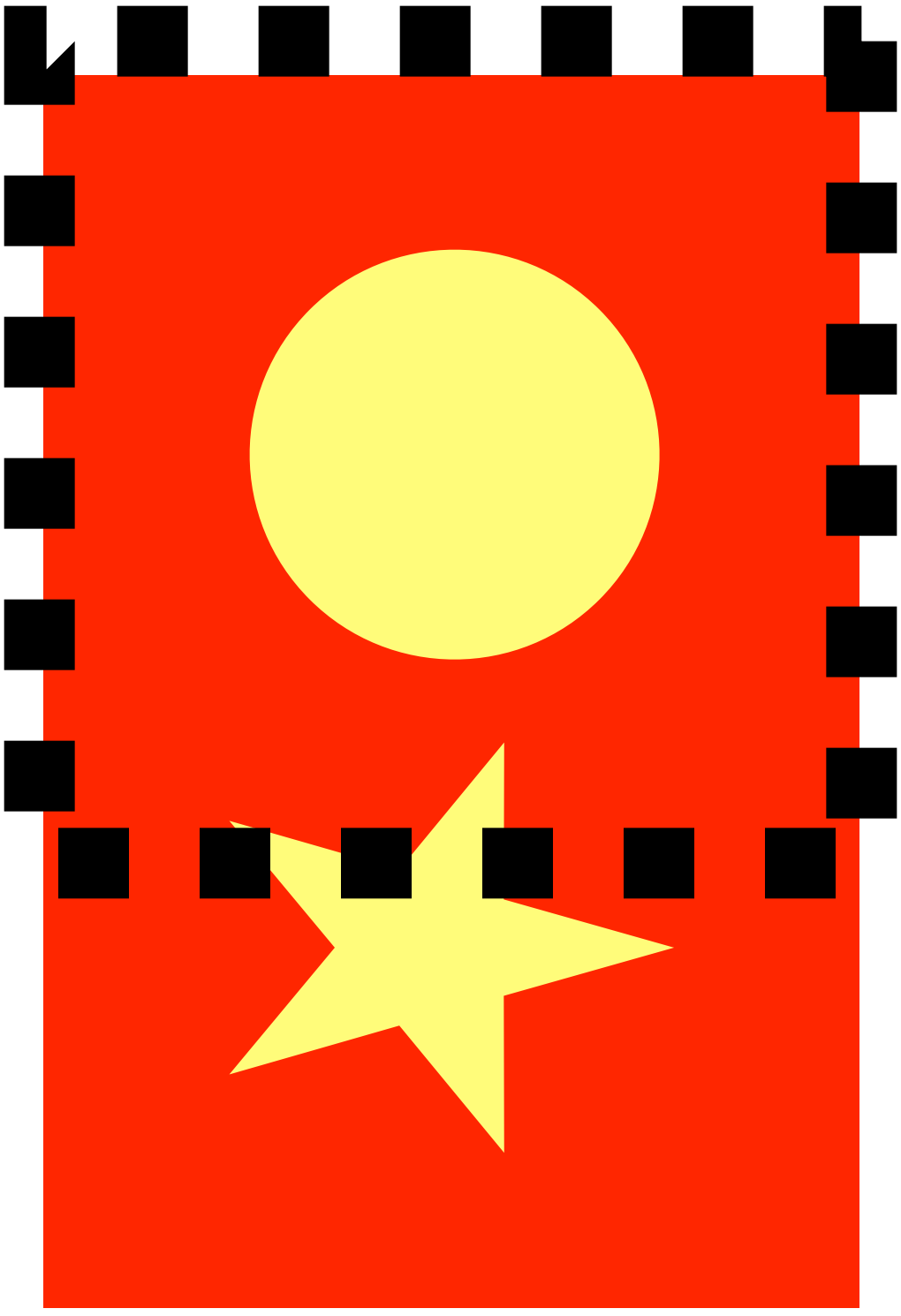
```
div { background-size: contain; }
```



cover keyword
value

The **cover keyword value** will scale the image (while keeping it’s aspect ratio) to the smallest size that will completely covered the background area.

```
div { background-size: cover; }
```



Shorthand backgrounds

The shorthand background property allows us to set all of the **individual background properties** using one rule.

The five CSS2.1 background properties could be written in **any order**.

```
p
{
    background:
        [background-color]
        [background-image]
        [background-repeat]
        [background-attachment]
        [background-position]
    ;
}
```

In CSS3, the eight background properties **have to be written in a specific order** - particularly background-size which is applied directly after background-position.

```
p
{
    background:
        [background-image]
        [background-position]
        [/ background-size]
        [background-repeat]
        [background-attachment]
        [background-origin]
        [background-clip]
        [background-color]
    ;
}
```

Multiple backgrounds

With CSS2.1, we could only apply **one background image** to any HTML element.

However, with CSS3 we can add **multiple background images** to any HTML element.

Longhand multiple backgrounds

CSS3 allows us to place **multiple, comma-separated values** into any background property.

```
p
{
    background-image:
        url(01.gif),
        url(02.gif),
        url(03.gif);
}
```

An example of **three background images** inside one element - written in longhand:

```
p
{
  background-image:       url(01.gif),   url(02.gif),   url(03.gif)  ;
  background-position:    left top,      50% 30%,       10px 100px   ;
  background-size:        auto,          10% auto,      auto         ;
  background-repeat:      no-repeat,     repeat,        repeat-y     ;
  background-attachment:  scroll,        scroll,        scroll       ;
  background-origin:      padding-box,   padding-box,   border-box   ;
  background-clip:        border-box,    padding-box,   border-box   ;
}
```

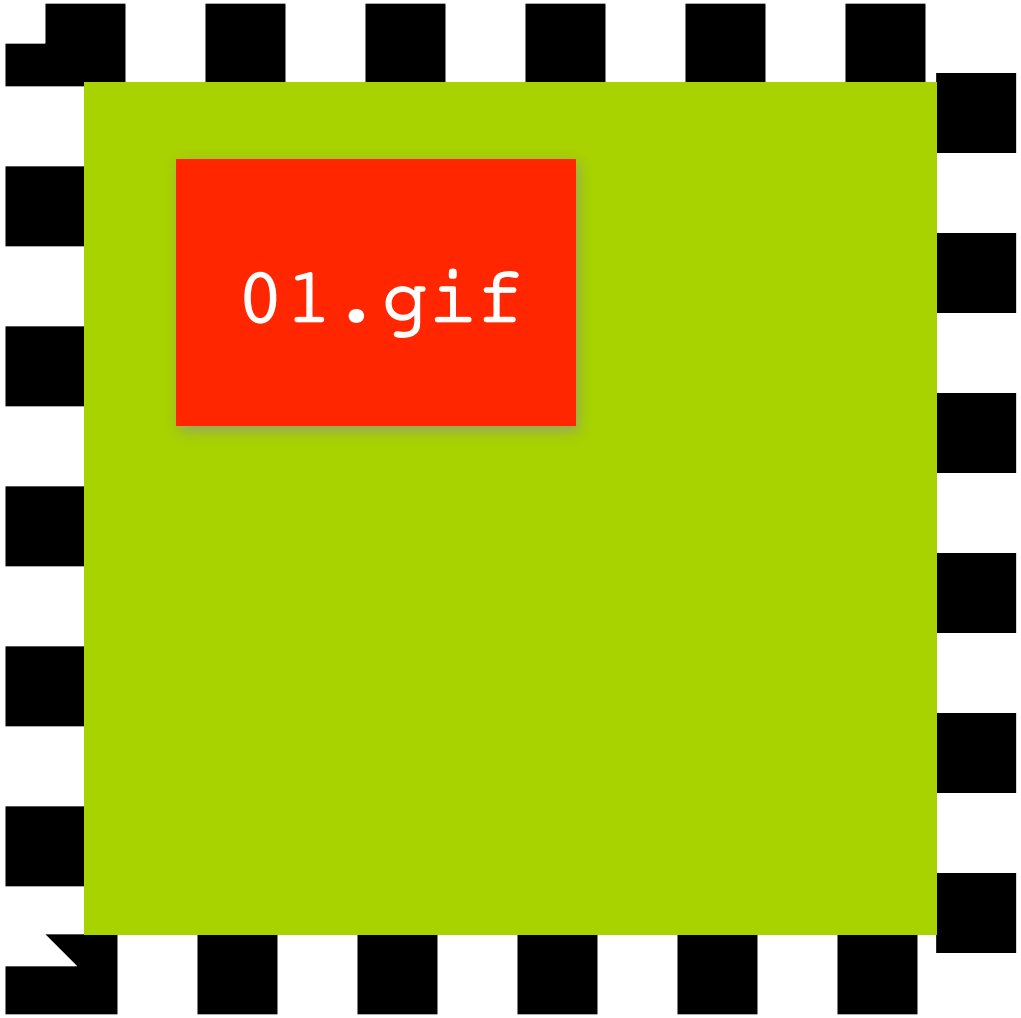
Each of the images is sized, positioned, and tiled according to the **corresponding value** in the other background properties.

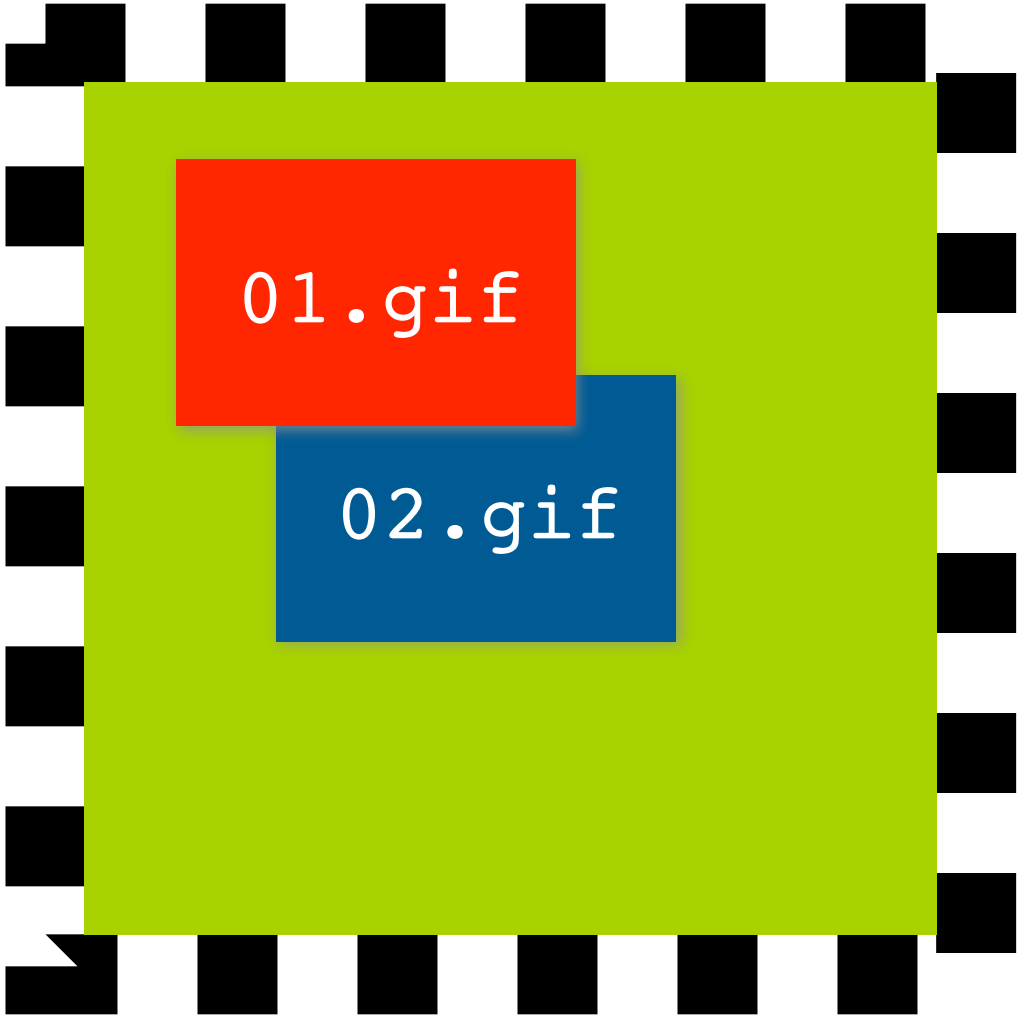
If a property doesn't have enough comma-separated values to match the number of layers, the browser must calculate its used value by **repeating the list of values** until there are enough.

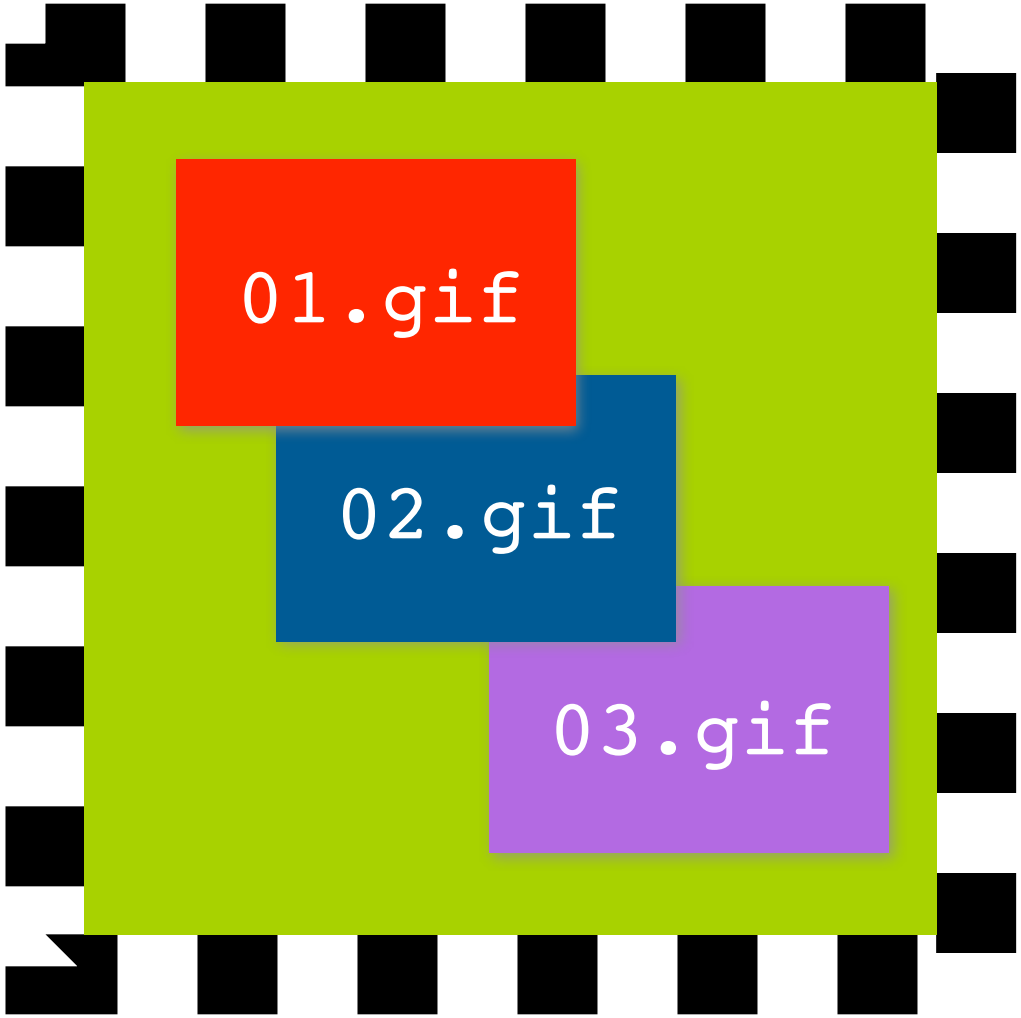
```
url(1.gif),   url(2.gif),   url(3.gif),    url(4.gif)
auto,         10% auto,     auto,          auto
no-repeat,    repeat,       [no-repeat],   [repeat]
```

The background images are then displayed in layers - one on top of each other. The first image in the list is the layer **closest to the user**, the next one is painted behind the first, and so on.

```
p
{
    background-image:
        url(01.gif),
        url(02.gif),
        url(03.gif);
}
```


01.gif


01.gif
02.gif


01.gif
02.gif
03.gif

Only **one background-color can be defined for any element**. This background-color sits below the background image on this final layer.

```
p
{
    background-image:
        url(01.gif),
        url(02.gif),
        url(03.gif);
    background-color: yellow;
}
```

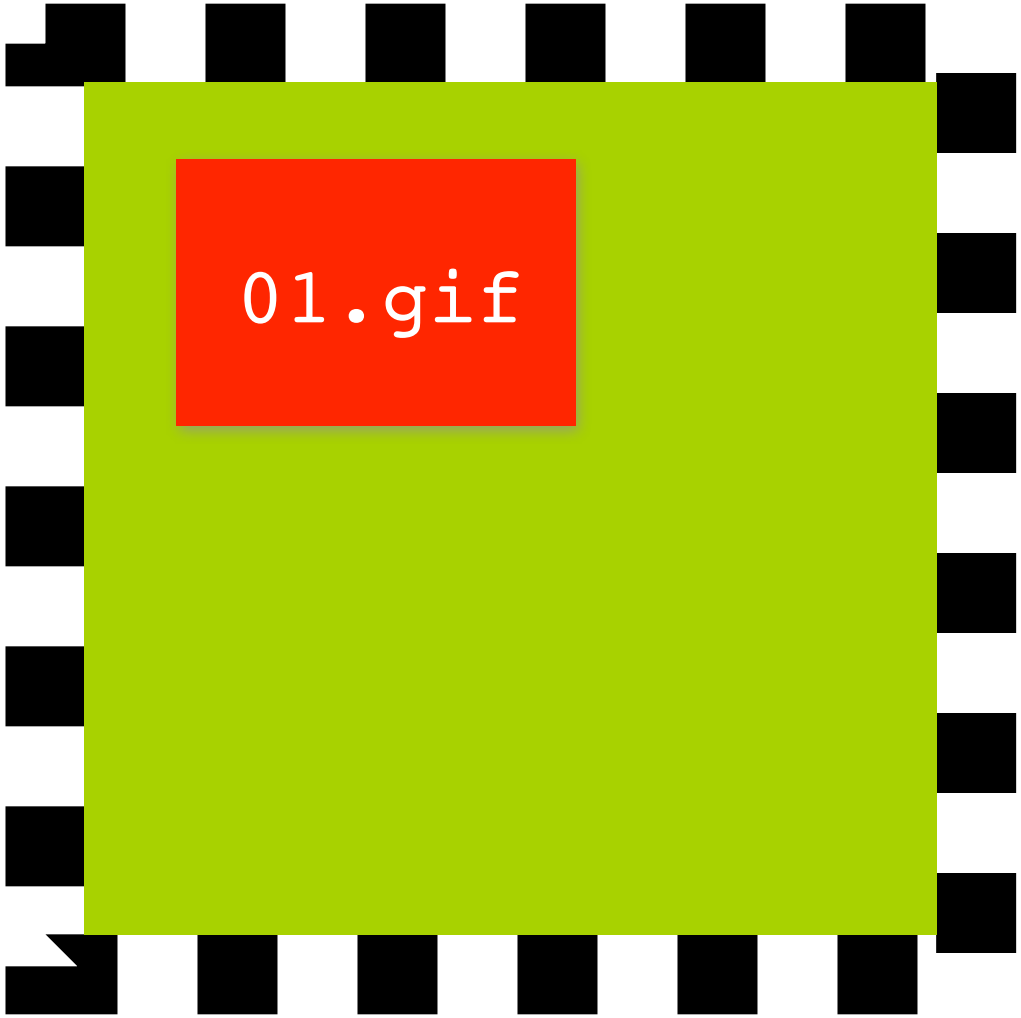
If more that one background-color value is assigned, **all background-colors will be ignored**.

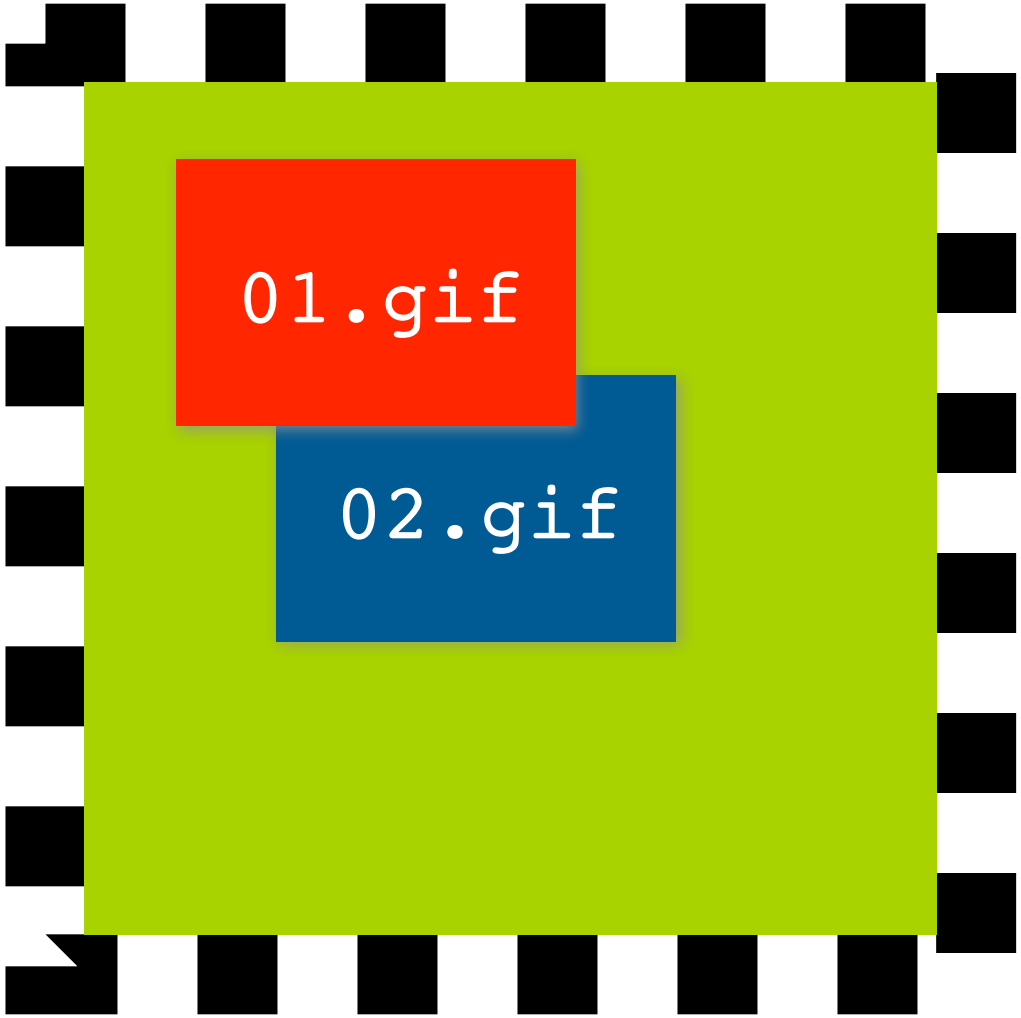
Shorthand multiple backgrounds

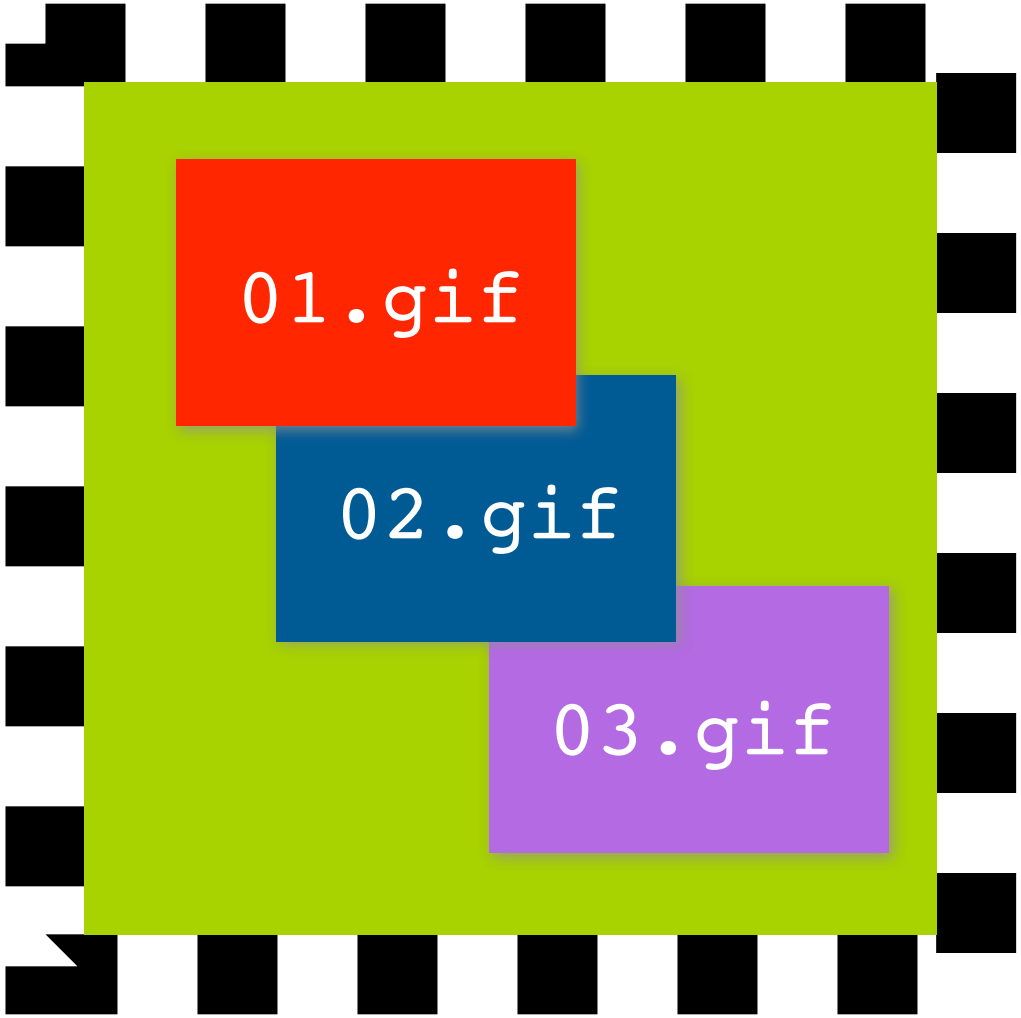
Multiple shorthand backgrounds are written in the same way as single shorthand backgrounds - with **comma's separating each background value**.

```
p
{
    background:
        url(01.gif) no-repeat,
        url(02.gif) repeat left bottom,
        url(03.gif) repeat-y 10px 5px;
}
```

Just as with the longhand version, the background images are displayed in layers - one on top of each other. The first image in the list is the layer **closest to the user**, the next one is painted behind the first, and so on.


01.gif


01.gif
02.gif


01.gif
02.gif
03.gif

Only the lowest layer, called the ‘final layer’, **can be given a background-color**. This background-color sits below the background images on this final layer only.

If a background-color value is assigned to any other layer apart from the final layer, **the entire rule will not be displayed**.

```
p
{
    background:
        url(01.gif) no-repeat,
        url(02.gif) repeat left bottom,
        url(03.gif) repeat-y 10px 5px yellow;
}
```

It may be safer to add the background-color as a **separate declaration** using the background-color property.

```
p
{
    background:
        url(01.gif) no-repeat,
        url(02.gif) repeat left bottom,
        url(03.gif) repeat-y 10px 5px;
    background-color: yellow;
}
```

Browser support

CSS3 Background-image options - CR

New properties to affect background images, including background-clip, background-origin and background-size

Global	88.53% + 7.91% =	96.44%
unprefixed:	88.53% + 7.75% =	96.29%

Current aligned | Usage relative | Show all

IE	Edge *	Firefox	Chrome	Safari	Opera	iOS Safari *	Opera Mini *	Android Browser *	Chrome for Android
								[3] 4.1	
8			43					[3] 4.3	
9		40	44					4.4	
10		41	45	8		8.4		4.4.4	
11	12	42	46	9	32	9.1	[1] 8	44	46
	13	43	47		33				
		44	48		34				
		45	49						

Russ Weakley

Max Design

Site: maxdesign.com.au

Twitter: twitter.com/russmaxdesign

Slideshare: slideshare.net/maxdesign

Linkedin: linkedin.com/in/russweakley