Forms Continued and Intro to CSS

Parameter Detail

`content-box` Width and height of the element only includes content area.

`padding-box` Width and height of the element includes content and padding.

`border-box` Width and height of the element includes content, padding and border.

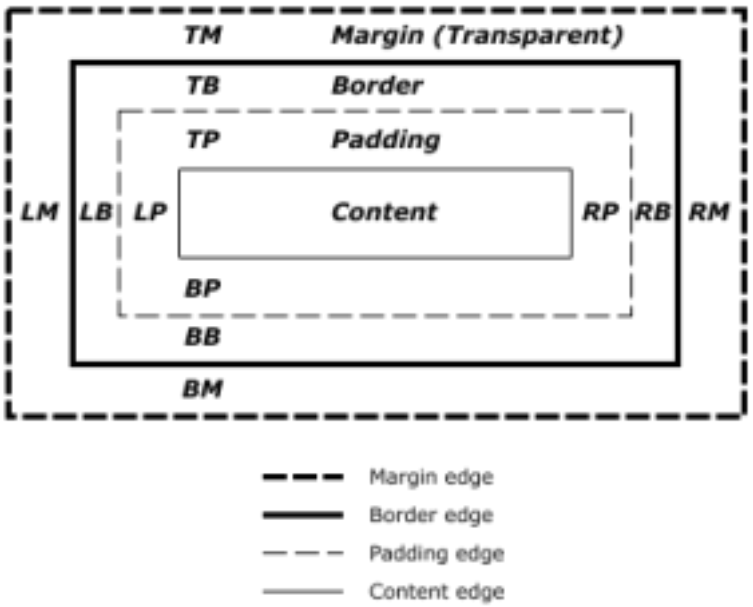
`initial` Sets the box model to its default state.

`inherit` Inherits the box model of the parent element.

What is the Box Model?

The Edges

The browser creates a rectangle for each element in the HTML document. The Box Model describes how the padding, border, and margin are added to the content to create this rectangle.



The perimeter of each of the four areas is called an *edge*. Each edge defines a *box.*

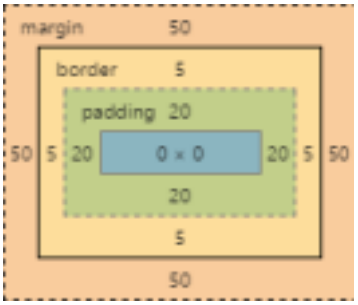
> The innermost rectangle is the **content box**. The width and height of this depends on the element's rendered content (text, images and any child elements it may have).
> Next is the **padding box**, as defined by the `padding` property. If there is no `padding` width defined, the padding edge is equal to the content edge.
> Then we have the **border box**, as defined by the `border` property. If there is no `border` width defined, the border edge is equal to the padding edge.

The outermost rectangle is the **margin box**, as defined by the `margin` property. If there is no `margin` width defined, the margin edge is equal to the border edge.

Example

```
div {
 border: 5px solid red;
 margin: 50px;
 padding: 20px;

}
```

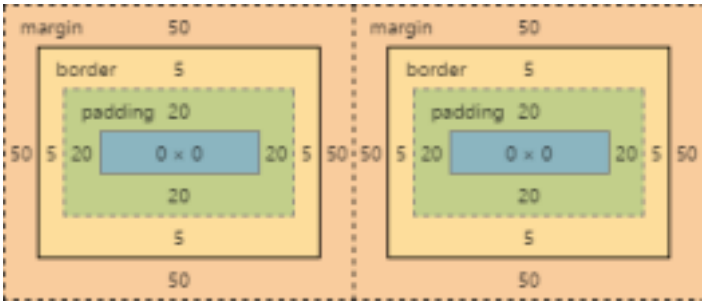

This CSS styles all `div` elements to have a top, right, bottom and left border of `5px` in width; a top, right, bottom and left margin of `50px`; and a top, right, bottom, and left padding of `20px`. Ignoring content, our generated box will look like this:



Screenshot of Google Chrome's Element Styles panel

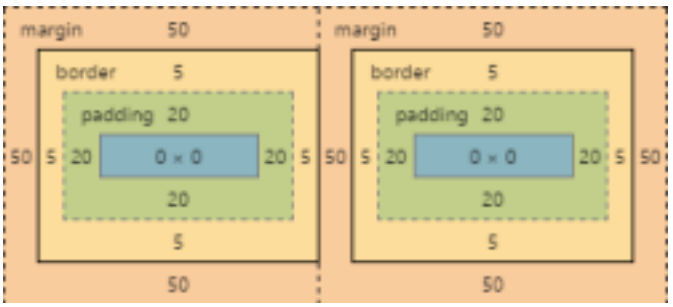
As there is no content, the content region (the blue box in the middle) has no height or width (0px by 0px).
The padding box by default is the same size as the content box, plus the 20px width on all four edges we're defining above with the `padding` property (40px by 40px).
The border box is the same size as the padding box, plus the 5px width we're defining above with the `border` property (50px by 50px).
Finally the margin box is the same size as the border box, plus the 50px width we're defining above with the `margin` property (giving our element a total size of 150px by 150px).

Now lets give our element a sibling with the same style. The browser looks at the Box Model of both elements to work out where in relation to the previous element's content the new element should be positioned:



The content of each of element is separated by a 150px gap, but the two elements' boxes touch each other.

If we then modify our first element to have no right margin, the right margin edge would be in the same position as the right border edge, and our two elements would now look like this:



box-sizing

The default box model (`content-box`) can be counter-intuitive, since the `width` / `height` for an element will not represent its actual width or height on screen as soon as you start adding `padding` and `border` styles to the

element.

The following example demonstrates this potential issue with `content-box`:

```
textarea {
 width: 100%;
 padding: 3px;
 box-sizing: content-box; /* default value */
}
```

Since the padding will be added to the width of the textarea, the resulting element is a textarea that is wider than 100%.

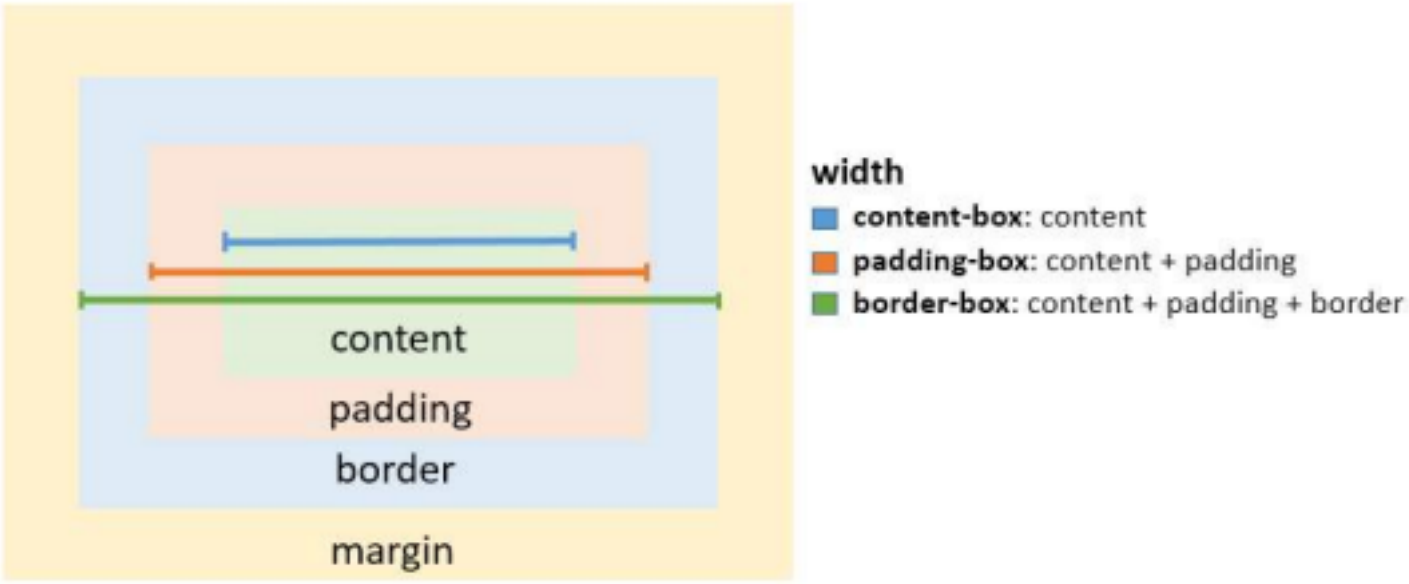
Fortunately, CSS allows us to change the box model with the `box-sizing` property for an element. There are three different values for the property available:

`content-box`: The common box model - width and height only includes the content, not the padding or border

`padding-box`: Width and height includes the content and the padding, but not the

border `border-box`: Width and height includes the content, the padding as well as the

border



To solve the `textarea` problem above, you could just change the `box-sizing` property to `padding-box` or `border box`. `border-box` is most commonly used.

```
textarea {
 width: 100%;
 padding: 3px;
 box-sizing: border-box;
}
```

To apply a specific box model to every element on the page, use the following snippet:

```
html {
 box-sizing: border-box;
}

*, *:before, *:after {
```



```
 box-sizing: inherit;
}
```

In this coding **`box-sizing`**`:border-box;` is not directly applied to `*`, so you can easily overwrite this property on individual elements.

Margins

Margin Collapsing

When two margins are touching each other vertically, they are collapsed. When two margins touch horizontally,

they do not collapse.

Example of adjacent vertical margins:

Consider the following styles and markup:

```
div{
 margin: 10px;
}
```

```
<div>
 some content
</div>
<div>
 some more content
</div>
```

They will be 10px apart since vertical margins collapse over one and other. (The spacing will not be the sum of two margins.)

Example of adjacent horizontal margins:

Consider the following styles and markup:

```
span{
 margin: 10px;
}
```

```
<span>some</span><span>content</span>
```

They will be 20px apart since horizontal margins don't collapse over one and other. (The spacing will be the sum of two margins.)

Overlapping with different sizes

```
.top{
 margin: 10px;
}
.bottom{
 margin: 15px;
}
```

```
<div class="top">
 some content



</div>
<div class="bottom">
 some more content
</div>
```

These elements will be spaced 15px apart vertically. The margins overlap as much as they can, but the larger margin will determine the spacing between the elements.

Overlapping margin gotcha

```
.outer-top{
 margin: 10px;
}
.inner-top{
```

```
 margin: 15px;
}
.outer-bottom{
 margin: 20px;
}
.inner-bottom{
 margin: 25px;
}
```

```
<div class="outer-top">
 <div class="inner-top">
 some content
 </div>
</div>
<div class="outer-bottom">
 <div class="inner-bottom">
 some more content
 </div>
</div>
```

What will be the spacing between the two texts? (hover to see answer)

> The spacing will be 25px. Since all four margins are touching each other, they will collapse, thus using the largest margin of the four.

Now, what about if we add some borders to the markup above.

```
div{
 border: 1px solid red;
}
```

What will be the spacing between the two texts? (hover to see answer)

> The spacing will be 59px! Now only the margins of .outer-top and .outer-bottom touch each other, and are the only collapsed margins. The remaining margins are separated by the borders. So we have 1px + 10px + 1px + 15px + 20px + 1px + 25px + 1px. (The 1px's are the borders...)

Collapsing Margins Between Parent and Child Elements:

HTML:

```
<h1>Title</h1>
<div>
 <p>Paragraph</p>
</div>
```

CSS

```
h1 {
 margin: 0;
 background: #cff;
}
div {
 margin: 50px 0 0 0;
 background: #cfc;
}
p {
 margin: 25px 0 0 0;
```

```
  background: #cf9;
}
```

In the example above, only the largest margin applies. You may have expected that the paragraph would be located 60px from the h1 (since the div element has a margin-top of 40px and the p has a 20px margin-top). This does not happen because the margins collapse together to form one margin.

Apply Margin on a Given Side

Direction-Specific Properties

CSS allows you to specify a given side to apply margin to. The four properties provided for this purpose are:

- `margin-left`
- `margin-right`
- `margin-top`
- `margin-bottom`

The following code would apply a margin of 30 pixels to the left side of the selected div. *View Result*

HTML

```
<div id="myDiv"></div>
```

CSS

```
#myDiv {
  margin-left: 30px;
  height: 40px;
  width: 40px;
  background-color: red;
}
```

Parameter Details

margin-left The direction in which the margin should be applied.

30px The width of the margin.

Specifying Direction Using Shorthand Property

The standard `margin` property can be expanded to specify differing widths to each side of the selected elements. The syntax for doing this is as follows:


```
margin: <top> <right> <bottom> <left>;
```

The following example applies a zero-width margin to the top of the div, a 10px margin to the right side, a 50px margin to the left side, and a 100px margin to the left side. *View Result*

HTML

```
<div id="myDiv"></div>
```

CSS

```
#myDiv {
  margin: 0 10px 50px 100px;
  height: 40px;
  width: 40px;
```

```
 background-color: red;
}
```

Margin property simplification

```
p {
 margin:1px; /* 1px margin in all directions */

 /*equals to:*/

 margin:1px 1px;

 /*equals to:*/

 margin:1px 1px 1px;

 /*equals to:*/

 margin:1px 1px 1px 1px;
}
```

Another exapmle:

```
p{
 margin:10px 15px; /* 10px margin-top & bottom And 15px margin-right & left*/
 /*equals to:*/

 margin:10px 15px 10px 15px;

 /*equals to:*/

 margin:10px 15px 10px;
 /* margin left will be calculated from the margin right value (=15px) */ }
```

Horizontally center elements on a page using margin

As long as the element is a **block**, and it has an **explicitly set width value**, margins can be used to center block elements on a page horizontally.

We add a width value that is lower than the width of the window and the auto property of margin then distributes the remaining space to the left and the right:

```
#myDiv {
 width:80%;
 margin:0 auto;
}
```

In the example above we use the shorthand `margin` declaration to first set `0` to the top and bottom margin values (although this could be any value) and then we use `auto` to let the browser allocate the space automatically to the left and right margin values.

In the example above, the #myDiv element is set to 80% width which leaves use 20% leftover. The browser distributes this value to the remaining sides so:

(100% - 80%) / 2 = 10%

Example 1:

It is obvious to assume that the percentage value of margin to `margin-left` and `margin-right` would be relative to its parent element.

```
.parent {
 width : 500px;
 height: 300px;
}

.child {
 width : 100px;
 height: 100px;
 margin-left: 10%; /* (parentWidth * 10/100) => 50px */
}
```

But that is not the case, when comes to `margin-top` and `margin-bottom`. Both these properties, in percentages, aren't relative to the height of the parent container but to the **width** of the parent container.

So,

```
.parent {
 width : 500px;
 height: 300px;
}

.child {
 width : 100px;
 height: 100px;
 margin-left: 10%; /* (parentWidth * 10/100) => 50px */
 margin-top: 20%; /* (parentWidth * 20/100) => 100px */
}
```

Negative margins

Margin is one of a few CSS properties that can be set to negative values. This property can be used to **overlap elements without absolute positioning**.

```
div{
```



```
 display: inline;
}

#over{
 margin-left: -20px;
}

<div>Base div</div>
<div id="over">Overlapping div</div>
```

Padding

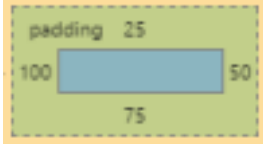
Padding Shorthand

The padding property sets the padding space on all sides of an element. The padding area is the space between the content of the element and its border. Negative values are not allowed.

To save adding padding to each side individually (using `padding-top`, `padding-left` etc) can you write it as a shorthand, as below:

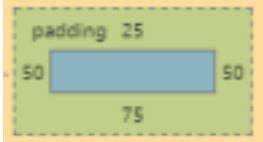
Four values:

```
<style>
 .myDiv {
 padding: 25px 50px 75px 100px; /* top right bottom left; */
 }
</style>
<div class="myDiv"></div>
```



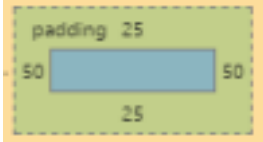
Three values:

```
<style>
 .myDiv {
 padding: 25px 50px 75px; /* top left/right bottom */
 }
</style>
<div class="myDiv"></div>
```



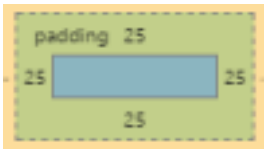
Two values:

```
<style>
 .myDiv {
 padding: 25px 50px; /* top/bottom left/right */
 }
</style>
<div class="myDiv"></div>
```



One value:

```
<style>
 .myDiv {
 padding: 25px; /* top/right/bottom/left */
 }
</style>
<div class="myDiv"></div>
```



Padding on a given side

The padding property sets the padding space on all sides of an element. The padding area is the space between the content of the element and its border. Negative values are not allowed.

You can specify a side individually:

- `padding-top`
- `padding-right`
- `padding-bottom`
- `padding-left`

The following code would add a padding of `5px` to the top of the div:

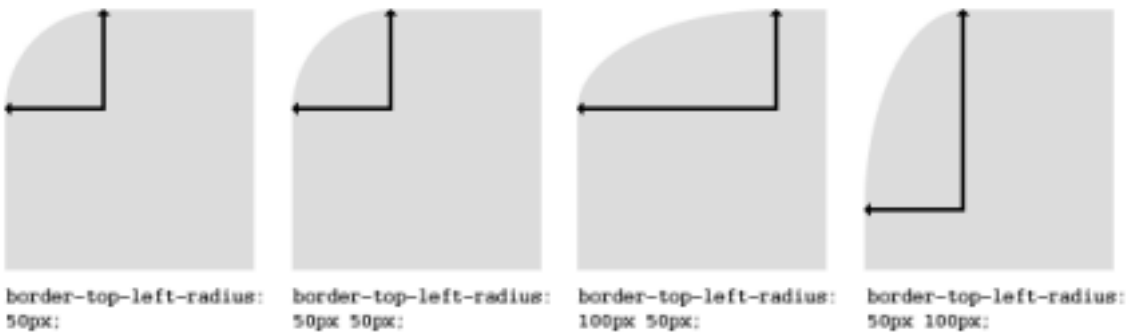
```
<style>
 .myClass {
  padding-top: 5px;
 }
</style>

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

Border

border-radius

The border-radius property allows you to change the shape of the basic box model.

Every corner of an element can have up to two values, for the vertical and horizontal radius of that corner (for a maximum of 8 values).



The first set of values defines the horizontal radius. The optional second set of values, preceded by a '/' , defines the vertical radius. If only one set of values is supplied, it is used for both the vertical and horizontal radius.

```
border-radius: 10px 5% / 20px 25em 30px 35em;
```

The `10px` is the horizontal radius of the top-left-and-bottom-right. And the `5%` is the horizontal radius of the

top right-and-bottom-left. The other four values after '/' are the vertical radii for top-left, top-right, bottom-right and bottom-left.

As with many CSS properties, shorthands can be used for any or all possible values. You can therefore specify anything from one to eight values. The following shorthand allows you to set the horizontal and vertical radius of every corner to the same value:

HTML:

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

CSS:

```
.box {
width: 250px;
height: 250px;
background-color: black;
border-radius: 10px;
}
```

Border-radius is most commonly used to convert box elements into circles. By setting the border-radius to half of the length of a square element, a circular element is created:

```
.circle {
width: 200px;
height: 200px;
border-radius: 100px;
}
```

Because border-radius accepts percentages, it is common to use 50% to avoid manually calculating the border radius value:

```
.circle {
width: 150px;
height: 150px;
border-radius: 50%;
}
```

If the width and height properties are not equal, the resulting shape will be an oval rather than a circle.

Browser specific border-radius example:

```
-webkit-border-top-right-radius: 4px;
-webkit-border-bottom-right-radius: 4px;
-webkit-border-bottom-left-radius: 0;
-webkit-border-top-left-radius: 0;
-moz-border-radius-topright: 4px;
-moz-border-radius-bottomright: 4px;
-moz-border-radius-bottomleft: 0;
-moz-border-radius-topleft: 0;
border-top-right-radius: 4px;
border-bottom-right-radius: 4px;
border-bottom-left-radius: 0;
border-top-left-radius: 0;
```

border-style

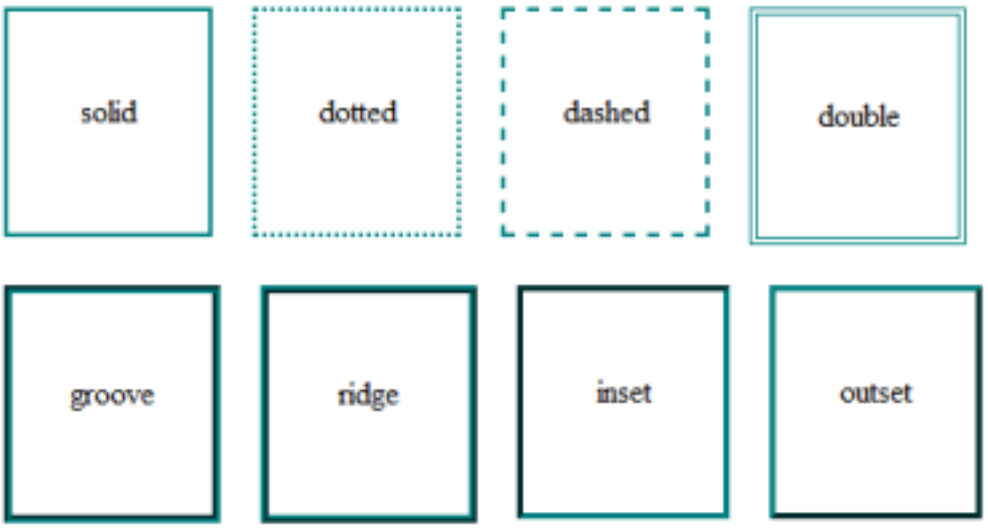
The `border-style` property sets the style of an element's border. This property can have from one to four

values (for every side of the element one value.)

Examples:

```
border-style: dotted;
```

```
border-style: dotted solid double dashed;
```



`border-style` can also have the values `none` and `hidden`. They have the same effect, except `hidden` works for border conflict resolution for `<table>` elements. In a `<table>` with multiple borders, `none` has the lowest priority (meaning in a conflict, the border would show), and `hidden` has the highest priority (meaning in a conflict, the border would not show).

Borders Using outline:

```
.div1{
border: 3px solid black;
outline: 6px solid blue;
width: 100px;
height: 100px;
margin: 20px;
}
```

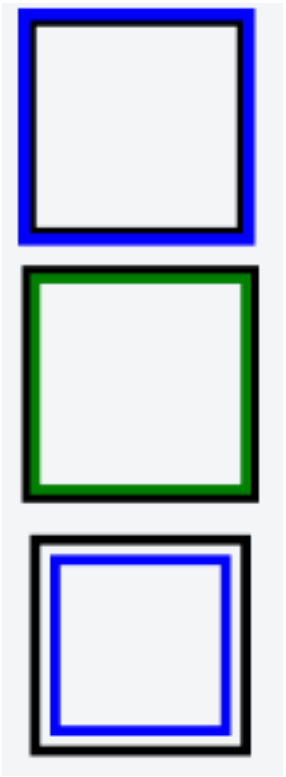
Using box-shadow:

```
.div2{
border: 5px solid green;
box-shadow: 0px 0px 0px 4px #000;
width: 100px;
height: 100px;
margin: 20px;
}
```

Using a pseudo element:

```
.div3 {
position: relative;
border: 5px solid #000;
width: 100px;
height: 100px;
margin: 20px;
}
```

```
.div3:before {
content: " ";
position: absolute;
border: 5px solid blue;
z-index: -1;
top: 5px;
left: 5px;
right: 5px;
bottom: 5px;
}
```



http://jsfiddle.net/MadalinaTn/bvqpcohm/2/

border (shorthands)

In most cases you want to define several border properties (`border-width`, `border-style` and `border-color`) for all sides of an element.

Instead of writing:

```
border-width: 1px;
border-style: solid;
border-color: #000;
```

You can simply write:

```
border: 1px solid #000;
```

These shorthands are also available for every side of an element: `border-top`, `border-left`, `border-right` and `border-bottom`. So you can do:

```
border-top: 2px double #aaaaaa;
```

border-collapse

The `border-collapse` property applies only to `table`s (and elements displayed as **`display`**: `table` or `inline`

`table`) and sets whether the table borders are collapsed into a single border or detached as in standard HTML.

```
table {
 border-collapse: separate; /* default */
 border-spacing: 2px; /* Only works if border-collapse is separate */
}
```

border-image

With the `border-image` property you have the possibility to set an image to be used instead of normal border styles.

A `border-image` essentially consist of a

- `border-image-source`: The path to the image to be used
- `border-image-slice`: Specifies the offset that is used to divide the image into **nine regions** (four **corners**, four **edges** and a **middle**)
- `border-image-repeat`: Specifies how the images for the sides and the middle of the border image are

scaled Consider the following example wheras border.png is a image of 90x90 pixels:

```
border-image: url("border.png") 30 stretch;
```

The image will be split into nine regions with 30x30 pixels. The edges will be used as the corners of the border while the side will be used in between. If the element is higher / wider than 30px this part of the image will be **stretched**. The middle part of the image defaults to be transparent.

border-[left|right|top|bottom]

The `border-[left|right|top|bottom]` property is used to add a border to a specific side of an

element. For example if you wanted to add a border to the left side of an element, you could do:

```
#element {
 border-left: 1px solid black;
}
```