CS 50 Web Design

APCSP Module 2: Internet

Unit 2: CSS (Chapter 14-15)

LECTURE 5: BOX MODEL AND FLOAT DESIGN

DR. ERIC CHOU

IEEE SENIOR MEMBER

Objectives

- The parts of an element box
- Setting box dimensions
- Padding
- Borders
- Outlines
- Margins
- Assigning display roles
- Adding a drop shadow

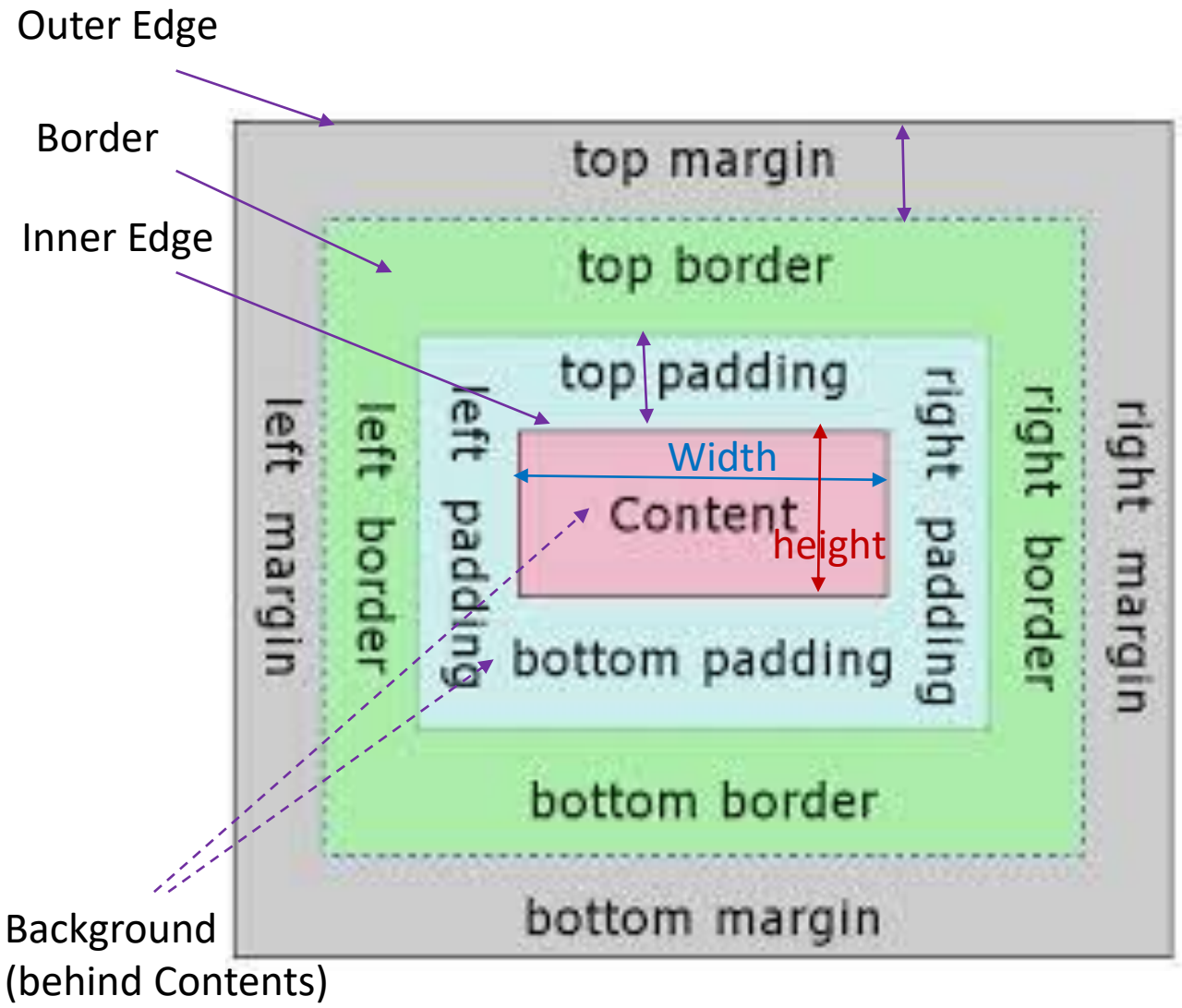
Objectives

- Floating elements to the
- left and right
- Clearing floated elements
- Containing floated elements
- Creating text-wrap shapes
- Relative positioning
- Absolute positioning and
- containing blocks
- Fixed positioning

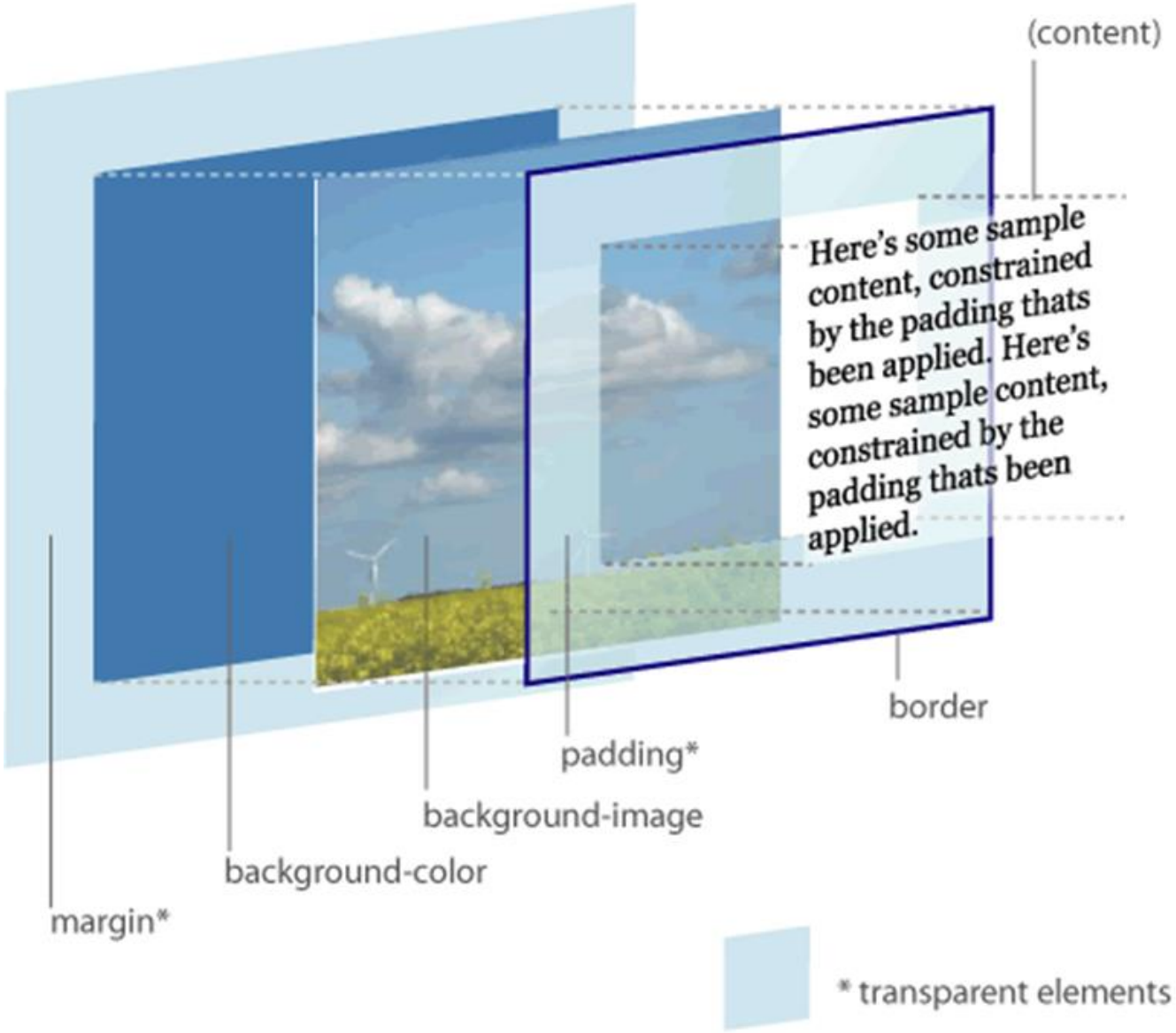
Overview of Box Model

SECTION 1

CSS3 Box Model



THE CSS BOX MODEL HIERARCHY



The Element Box

Content area

At the core of the element box is the content itself. In FIGURE 14-1, the content area is indicated by a white box.

Inner edges

The edges of the content area are referred to as the inner edges of the element box. Although the inner edges are made distinct by a color change in FIGURE 14-1, in real pages, the edge of the content area is invisible.

Padding

The padding is the area between the content area and an optional border. In the diagram, the padding area is indicated by a yellow-orange color. Padding is optional.

The Element Box

Border

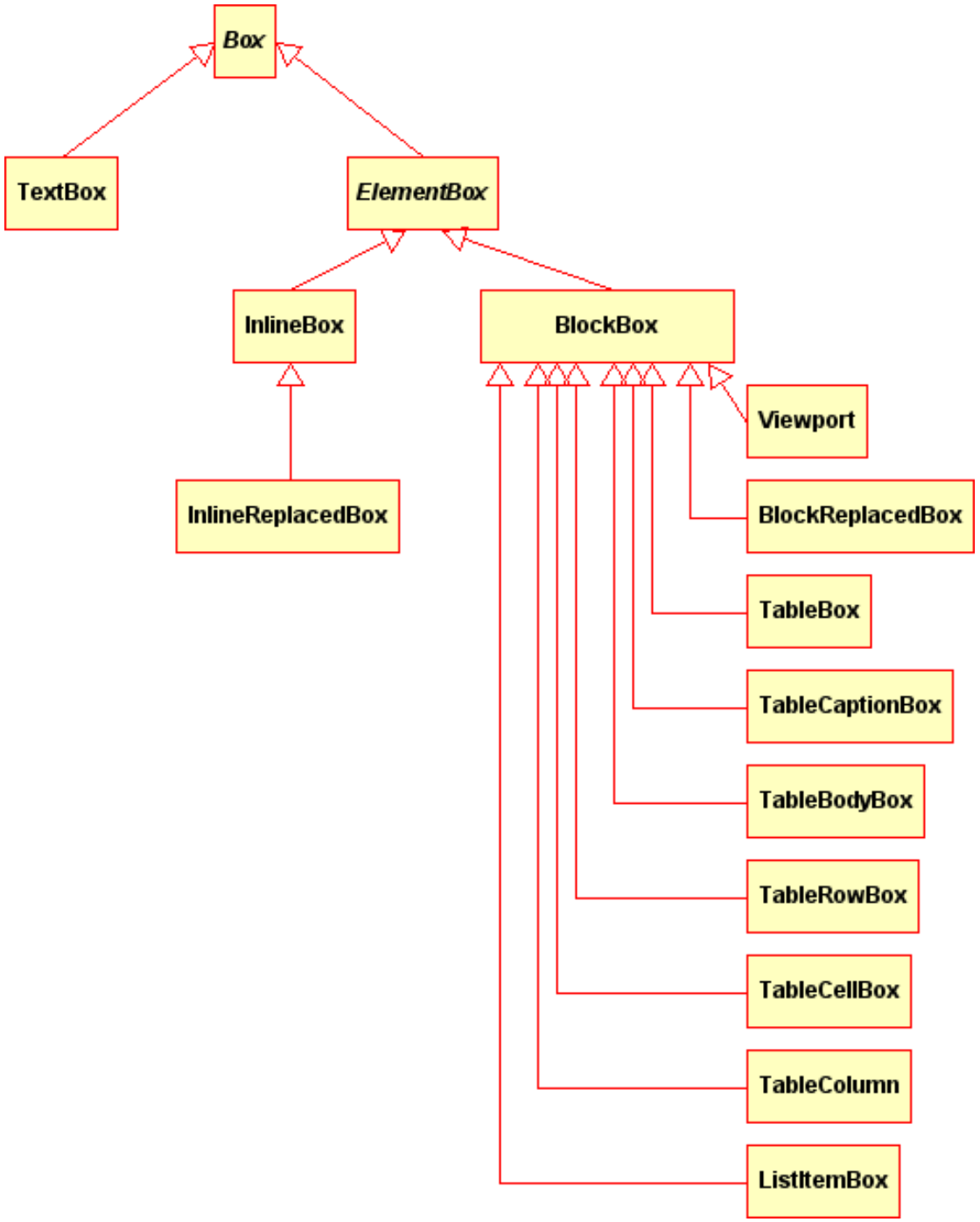
The border is a line (or stylized line) that surrounds the element and its padding. Borders are also optional.

Margin

The margin is an optional amount of space added on the *outside* of the border. In the diagram, the margin is indicated with light-blue shading, but in reality, margins are always transparent, allowing the background of the parent element to show through.

Outer edge

The outside edges of the margin area make up the outer edges of the element box. This is the total area the element takes up on the page, and it includes the width of the content area plus the total amount of padding, border, and margins applied to the element. The outer edge in the diagram is indicated with a dotted line, but in real web pages, the edge of the margin is invisible.



Box Classes

Box model applies to all of the elements.

- Text
- Table
- Image
- List Item
- Form

And many others.

width

Values:	*length* | *percentage* | auto
Default:	auto
Applies to:	block-level elements and replaced inline elements (such as images)
Inherits:	no

height

Values:	*length* | *percentage* | auto
Default:	auto
Applies to:	block-level elements and replaced inline elements (such as images)
Inherits:	no

box-sizing

Values:	content-box | border-box
Default:	content-box
Applies to:	all elements
Inherits:	no

Specifying Box Dimensions

Element box =

20px + 5px + 20px + 500px width + 20px + 5px + 20px = 590 pixels

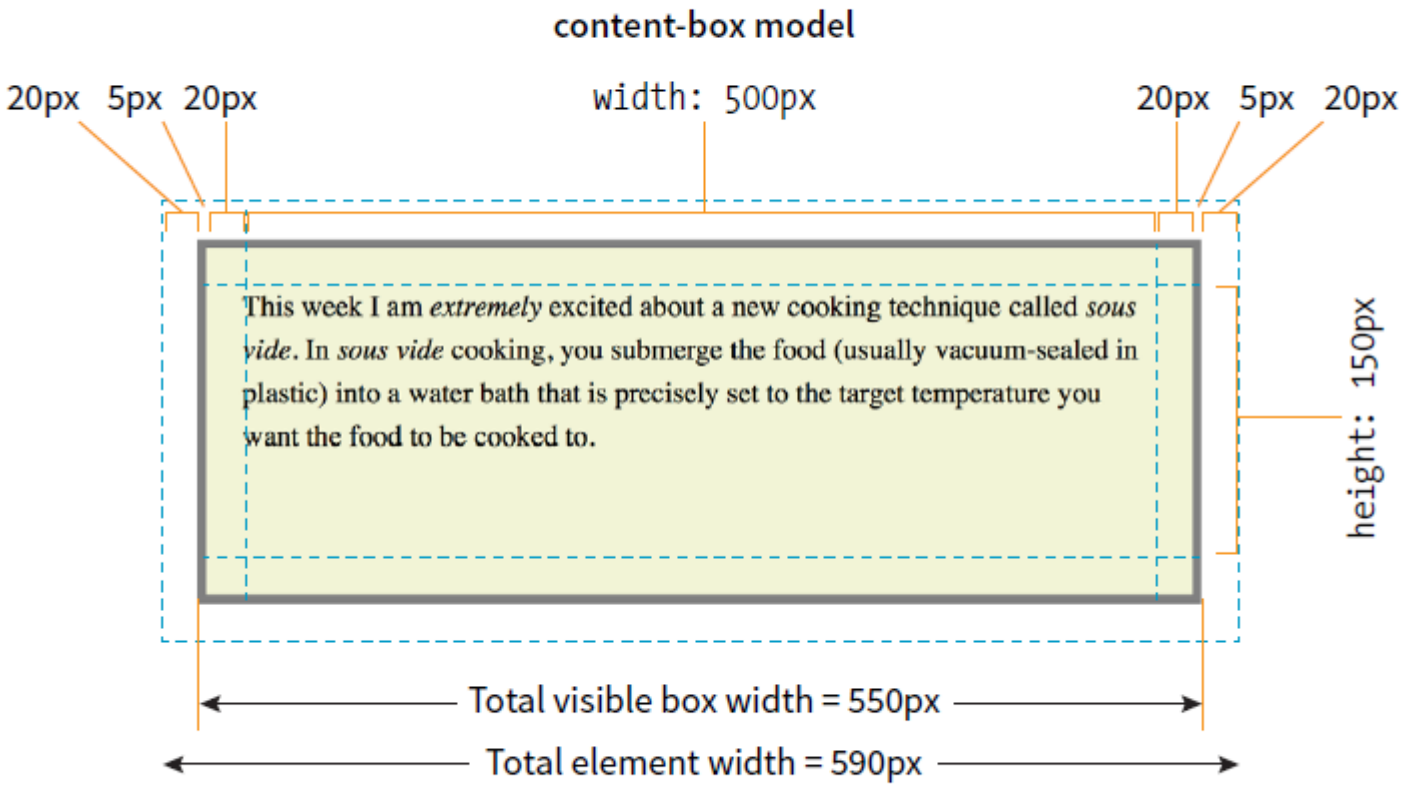


FIGURE 14-2. Specifying the **width** and **height** with the **content-box** model.

Sizing the Content Box

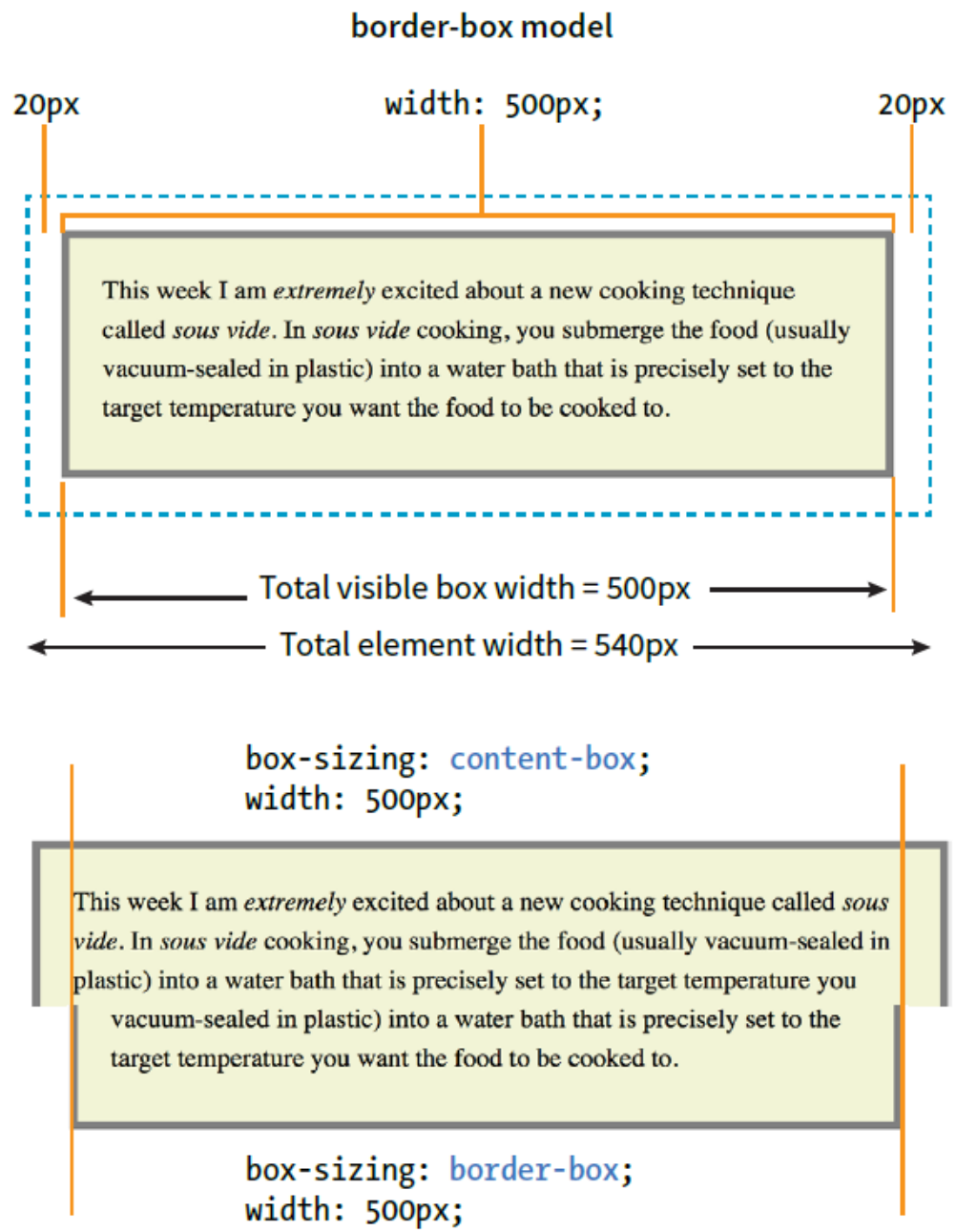


FIGURE 14-3. Sizing an element with the **border-box** method. The bottom diagram compares the resulting boxes from each sizing method.

Using the border-box Model

```
p {
    …
    box-sizing: border-box;
    width: 500px;
    height: 150px;
}
```

Maximum and Minimum Dimensions

- you want to set a limit on the size of a block element, use the **max-** and **min-** width and height properties.
  max-height, **max-width**,
  min-height, **min-width**
  Values: *length* | *percentage* | none

- These properties work with blocklevel and replaced elements (like images) only. When the **contentbox** model is used, the value applies to the content area only, so if you apply padding, borders, or margins, it will make the overall element box larger, even if a **max-width** or **maxheight** property has been specified.
- Note also that IE8 does not support **box-sizing** on elements with **max-**/ **min-** sizes.

Specifying Height

Handling Overflow

When an element is sized too small for its contents, you can specify what to do with the content that doesn't fit by using the **overflow** property.

overflow

Values:	visible | hidden | scroll | auto
Default:	visible
Applies to:	block-level elements and replaced inline elements (such as images)
Inherits:	no

visible

The default value is **visible**, which allows the content to hang out over the element box so that it all can be seen.

hidden

When **overflow** is set to **hidden**, the content that does not fit gets clipped off and does not appear beyond the edges of the element's content area.

scroll

When **scroll** is specified, scrollbars are added to the element box to let users scroll through the content. Be aware that they may become visible only when you click the element to scroll it. There is an issue with this value on old iOS (<4), Android (<2.3), and a few other older mobile browsers, so it may be worthwhile to use a simpler alternative to **overflow:scroll** for mobile.

auto

The **auto** value allows the browser to decide how to handle overflow. In most cases, scrollbars are added only when the content doesn't fit and they are needed.

Specifying Height

Example of Content Box Sizing

```
p {
    background: #c2f670;
    width: 500px;
    height: 150px;
    padding:20px;
    border: 2px solid gray;
    margin 20px;
}
```

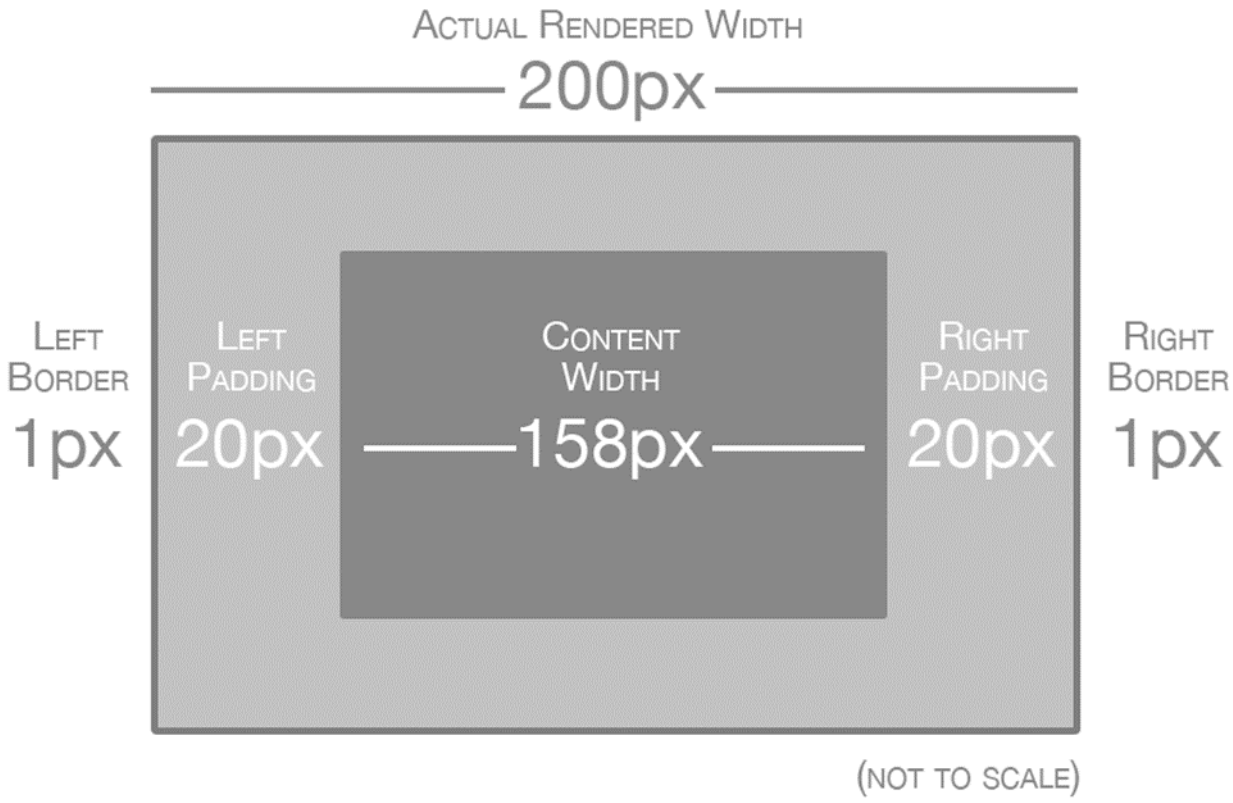
Total size for width: 20px + 2px + 20px + 500px + 20px + 2px + 20px = 584 px

Total visible box width = 2px + 20px + 500px + 20px + 2px = 544px

Total size for height: 20px + 2px + 20px + 150px + 20px + 2px + 20px = 234px

Total visible box height = 2px + 20px + 150px +20px + 2px = 194px

```
.sidebar {
  width: 158px;
  padding: 20px;
  border: 1px solid #DDD;
}
```



Content Box Sizing

Example of Border Box Sizing

```
p {
    box-sizing: border-box
    background: #c2f670;
    width: 500px;
    height: 150px;
    padding:20px;
    border: 2px solid gray;
    margin 20px;
}
```

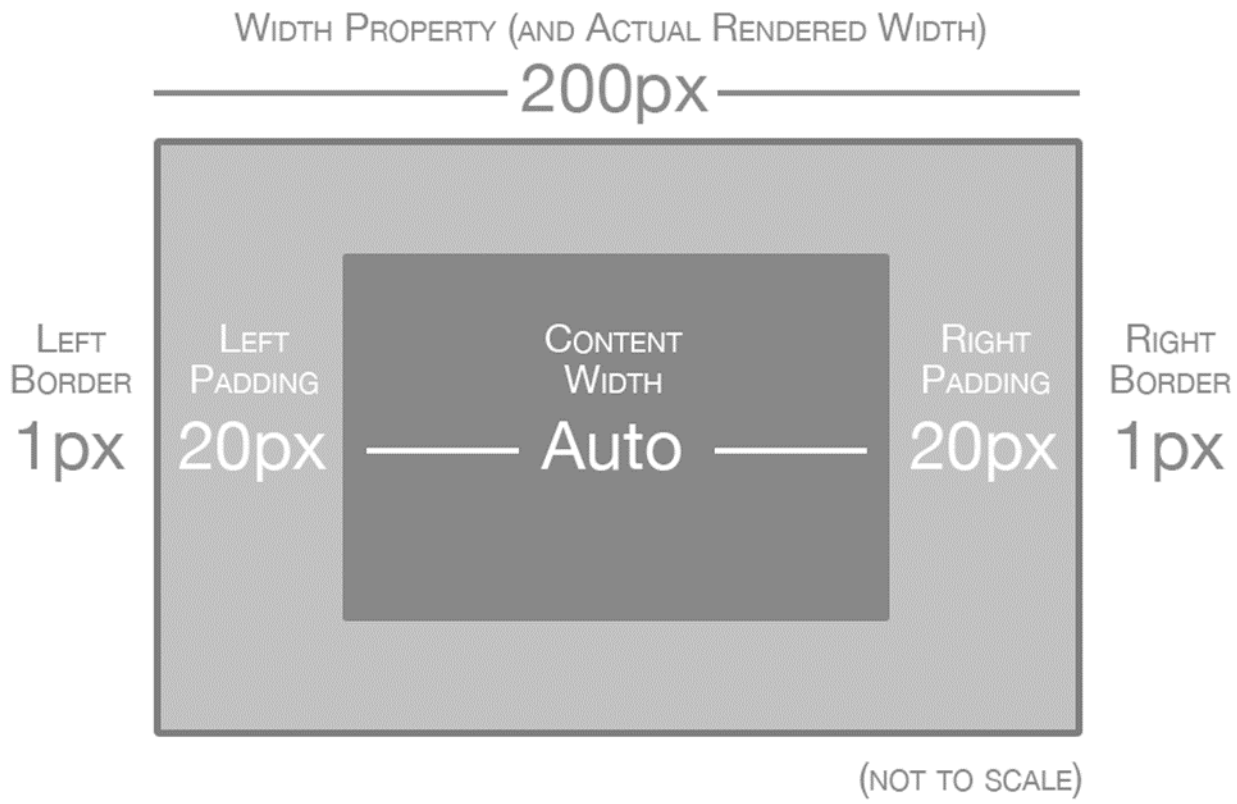
Total size for width: 20px + 500px + 20px = 540px

Total content box width = -2px - 20px + 500px - 20px - 2px = 456px

Total size for height: 20px + 150px + 20px = 234px

Total content box height = -2px - 20px + 150px -20px - 2px = 116px

```
.sidebar {
  box-sizing: border-box;
  width: 200px;
  padding: 20px;
  border: 1px solid #DDD;
}
```



Border Box Sizing

Overflow handling

Overflow: **visible | hidden | scroll | auto | inherit**

visible: visible is default value, which allows the content to hang out over the element box so that it all can be seen.

hidden: when overflow is set to hidden , the content that does not fit gets clipped off and does not appear beyond the edges of the element's content area.

scroll: When scroll is specified, scrollbars are added to the element box to let users scroll through the content. Be aware that when you set the value to scroll, the scrollbars will always be there, even if the content fits in the specified height just fine.

auto: the auto value allows the browser to decide how to handle overflow. In most cases, scrollbars are added only when the content doesn't fit and they are needed.

Padding

SECTION 2

Padding

- Padding is the space between the content area and the border (or the place the border would be if one isn't specified). I find it helpful to add padding to elements when using a background color or a border. It gives the content a little breathing room, and prevents the border or edge of the background from bumping right up against the text.
- You can add padding to the individual sides of any element (block-level or inline). There is also a shorthand **padding** property that lets you add padding on all sides at once.

Padding

padding-top, **padding-right**, **padding-bottom**, **padding-left**

Values: *length* | *percentage*
Default: 0
Applies to: all elements
Inherits: no

padding

Values: *length* | *percentage*
Default: 0
Applies to: all elements
Inherits: no

```
blockquote {
  padding-top: 2em;
  padding-right: 4em;
  padding-bottom: 2em;
  padding-left: 4em;
  background-color: #D098D4; /* light green */
}
```

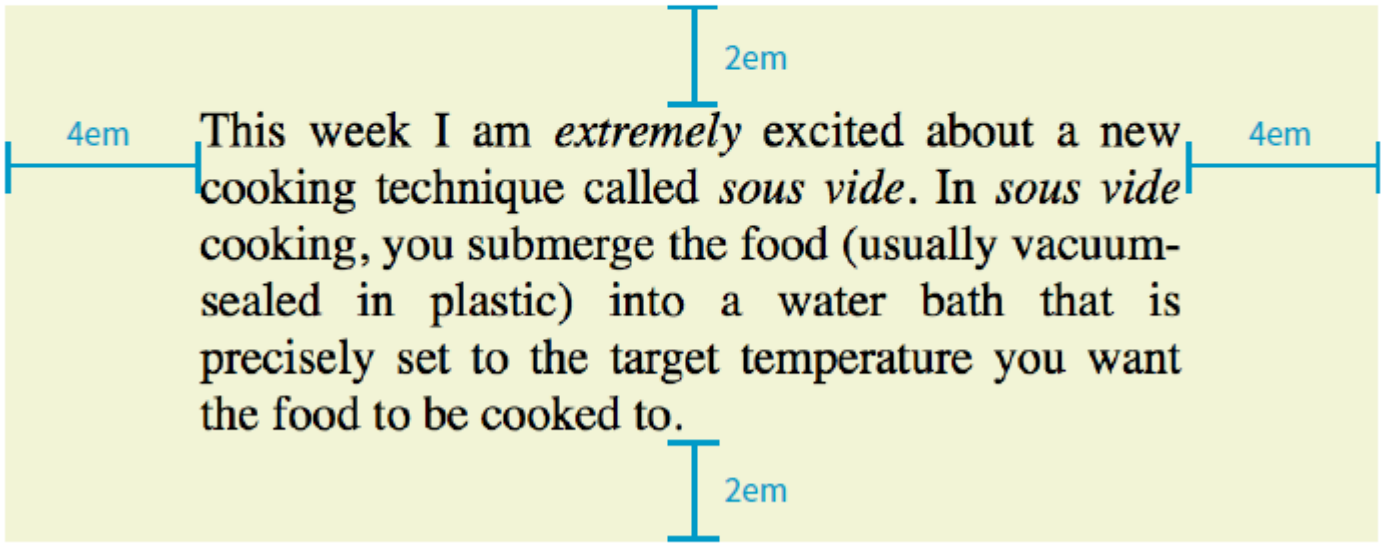


FIGURE 14-5. Adding padding around the content of an element.

Padding

Shorthand Values

1 value

padding: 10px;

Applied to all sides.

2 values

padding: 10px 6px;

First is top and bottom; second is left and right.

3 values

padding: 10px 6px 4px;

First is top; second is left and right; third is bottom.

4 values

padding: 10px 6px 4px 10px;

Applied clockwise to top, right, bottom, and left edges consecutively (TRBL).

Padding
Shorthand Values

Border

SECTION 3

Border

- A border is simply a line drawn around the content area and its (optional) padding. You can choose from eight border styles and make them any width and color you like. Borders can be applied all around the element or just on a particular side or sides.
- CSS3 introduced properties for rounding the corners or applying images to borders. We'll start our border exploration with the various border styles.

Border Style

border-top-style, **border-right-style**,
border-bottom-style, **border-left-style**

Values:	none | solid | hidden | dotted | dashed | double | groove | ridge | inset |outset
Default:	none
Applies to:	all elements
Inherits:	no

border-style

Values:	none | solid | hidden | dotted | dashed | double | groove | ridge | inset |outset
Default:	none
Applies to:	all elements
Inherits:	no

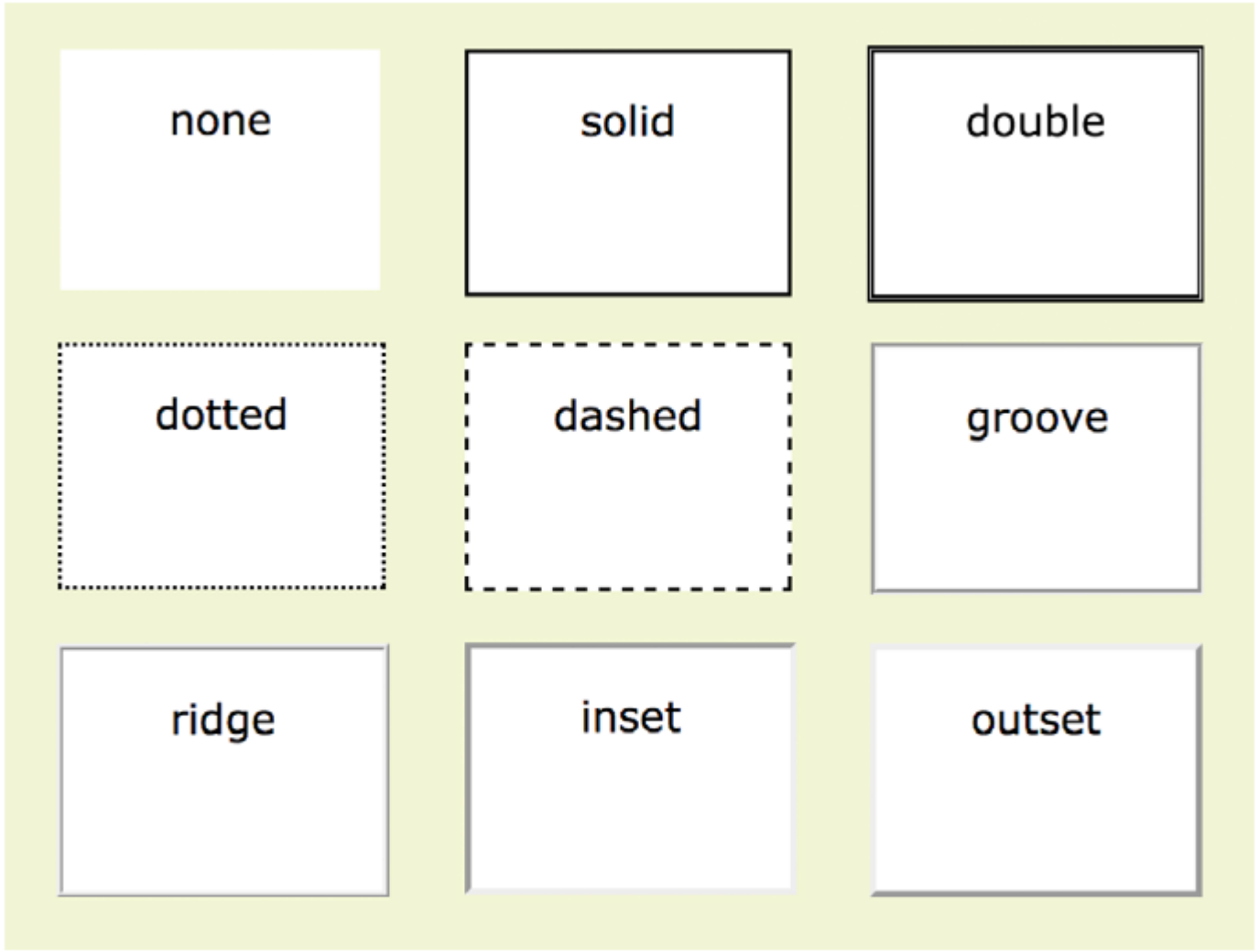


FIGURE 14-8. The available border styles (shown at the default medium width).

Border Style

```
div#silly {
    border-top-style: solid;
    border-right-style: dashed;
    border-bottom-style: double;
    border-left-style: dotted;
    width: 300px;
    height: 100px;
}
```

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

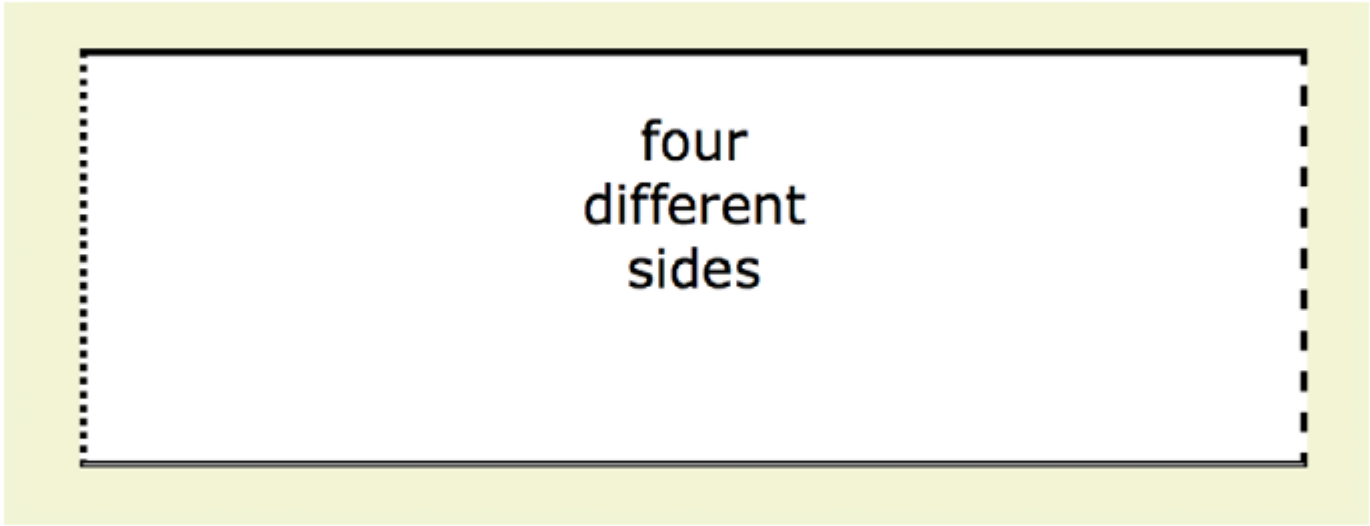


FIGURE 14-9. Border styles applied to individual sides of an element.

Border Style

Border Color

- Border colors are specified in the same way: via the side-specific properties or the **border-color** shorthand property. When you specify a border color, it overrides the foreground color as set by the **color** property for the element.

border-top-color, **border-right-color**, **border-bottom-color**, **border-left-color**

Values:	*color name* or *RGB/HSL value* | transparent
Default:	the value of the color property for the element
Applies to:	all elements
Inherits:	no

border-color

Values:	*color name* or *RGB/HSL value* | transparent
Default:	the value of the color property for the element
Applies to:	all elements
Inherits:	no

```
div#special {
  border-color: maroon aqua;
  border-style: solid;
  border-width: 6px;
  width: 300px;
  height: 100px;
}
```

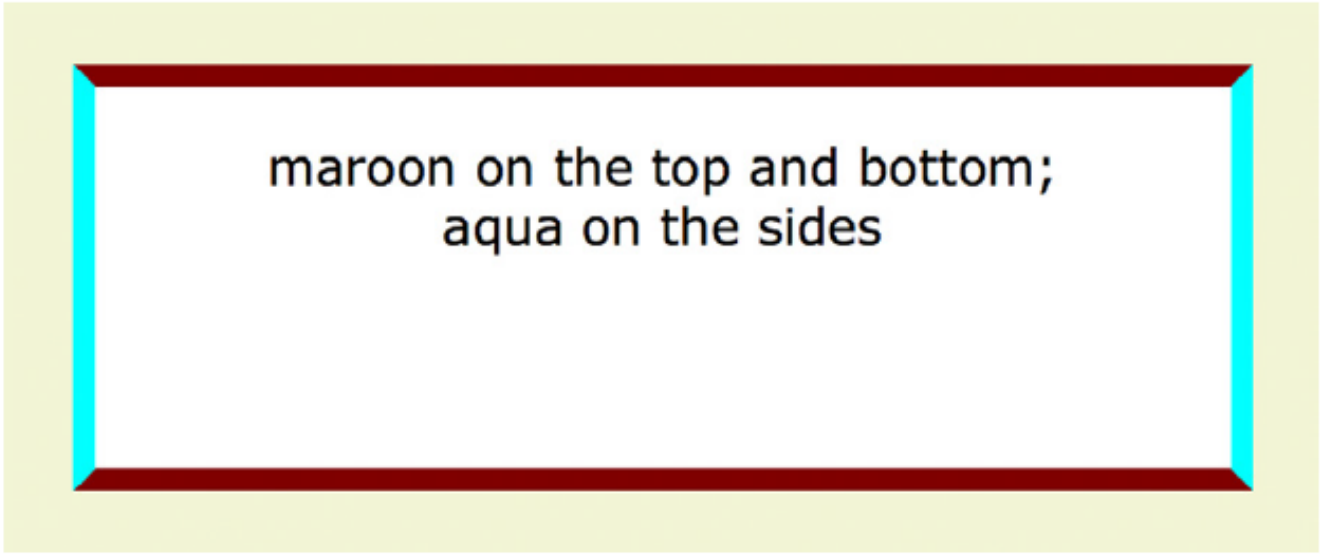


FIGURE 14-11. Specifying the color of borders.

Border Color

Combining Style, Width, and Color

- The authors of CSS didn't skimp when it came to border shortcuts. They also created properties for providing style, width, and color values in one declaration, one side at a time. You can specify the appearance of specific sides, or use the **border** property to change all four sides at once.

border-top, **border-right**, **border-bottom**, **border-left**

Values:	*border-style border-width border-color*
Default:	defaults for each property
Applies to:	all elements
Inherits:	no

border

Values:	*border-style border-width border-color*
Default:	defaults for each property
Applies to:	all elements
Inherits:	no

```
h1 { border-left: red .5em solid; }      /* left border only */
h2 { border-bottom: 1px solid; }         /* bottom border only */
p.example { border: 2px dotted #663; }   /* all four sides */
```

border-top-left-radius, **border-top-right-radius**, **border-bottom-right-radius**, **border-bottom-left-radius**

Values:	*length* | *percentage*
Default:	0
Applies to:	all elements
Inherits:	no

border-radius

Values:	*1, 2, 3, or 4 length or percentage values*
Default:	0
Applies to:	all elements
Inherits:	no

Rounded Corners with border-radius

```
p {
    width: 200px;
    height: 100px;
    background: darkorange;
}
```

border-radius: 1em;

border-top-right-radius: 50px;

border-radius: 50px;

```
border-top-left-radius: 1em;
border-top-right-radius: 2em;
border-bottom-right-radius: 1em;
border-bottom-left: 2em;
```

FIGURE 14-12. Make the corners of element boxes rounded with the **border-radius** properties.

Rounded Corners with border-radius

Ⓓ `border-radius: 36px 40px 60px 20px / 12px 10px 30px 36px;`

Ⓐ

`border-top-right-radius: 100px 50px;`

Ⓑ

`border-top-right-radius: 50px 20px;`
`border-top-left-radius: 50px 20px;`

Ⓒ

`border-radius: 60px / 40px;`

Ⓓ

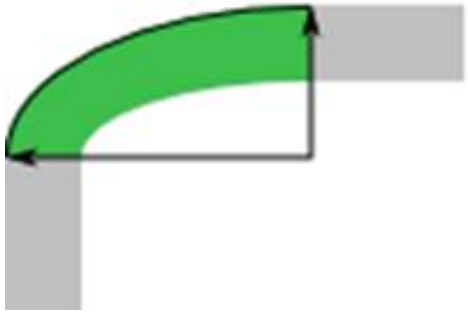
`border-radius:`
`36px 40px 60px 20px/12px 10px 30px 36px;`

FIGURE 14-13. Applying elliptical corners to boxes.

Elliptical corners

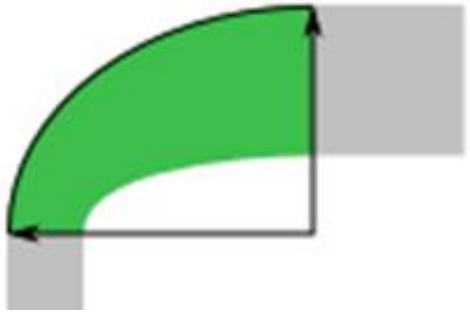
A

```
border-width: 20px;
border-radius:
80px / 40px;
```



B

```
border-width:
40px 20px;
border-radius:
80px / 60px;
```

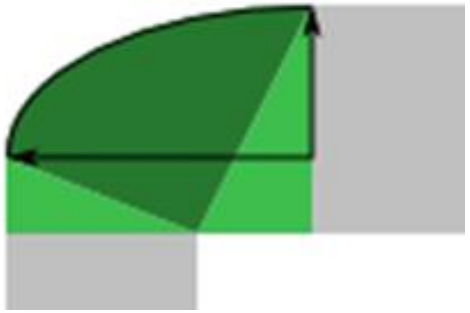


C

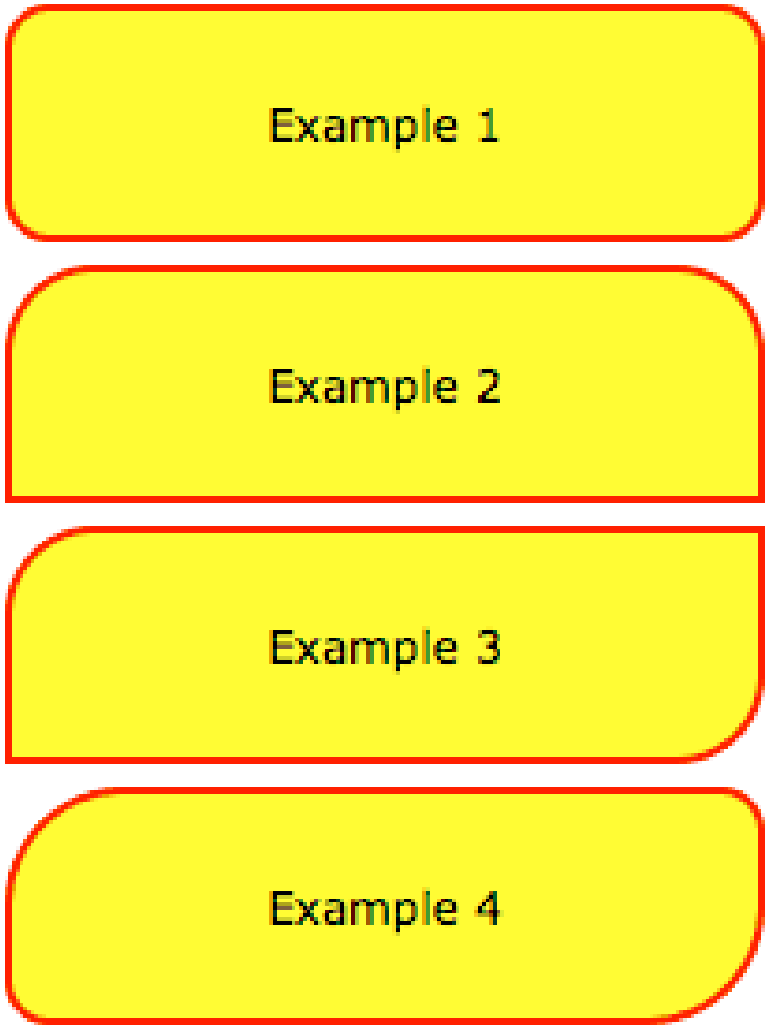
```
border-width:
40px 80px;
border-radius:
80px / 40px;
```

D

```
border-width:
60px 40px;
border-radius:
80px / 40px;
```



Radius and Width




```
-moz-border-radius: 1em;
```

```
-moz-border-radius-topright: 2em;
-moz-border-radius-topleft: 2em;
```

```
-moz-border-radius: 2em 0;
```

```
-moz-border-radius: 3em 1em;
```

Example 5

```
-webkit-border-radius: 1em;
```

Example 6

```
-webkit-border-top-right-radius: 24px;
-webkit-border-top-left-radius: 24px;
```

Example 7

```
-webkit-border-radius: 24px 0;
```

Example 8

```
-webkit-border-radius: 36px 12px;
```

Example 9

```
-webkit-border-top-right-radius: 40px 30px;
-webkit-border-bottom-right-radius: 40px 30px;
```

Picture-Perfect Borders

Border images are applied with a collection of five properties:

- **border-image-source** indicates the location of the image
- **border-image-slice** divides the image into nine sections using offset measurements
- **border-image-width** specifies the width of the border area
- **border-image-repeat** specifies whether the image should stretch or repeat along the sides
- **border-image-outset** pushes the border away from the content by the specified amount

Picture-Perfect Borders

```
border: 5px solid #d1214a; /* red */
border-image: url(fancyframe.png) 55 fill / 55px /
25px stretch;
```

FIGURE 14-15. Examples of a border image applied to a box.

Margins

SECTION 4

Margins

margin-top, **margin-right**, **margin-bottom**, **margin-left**

Values:	*length* | *percentage* | auto
Default:	auto
Applies to:	all elements
Inherits:	no

margin

Values:	*length* | *percentage* | auto
Default:	auto
Applies to:	all elements
Inherits:	no

```
Ⓐ p#A {
    margin: 4em;
    border: 2px solid red;
    background: #e2f3f5;
  }
Ⓑ p#B {
    margin-top: 2em;
    margin-right: 250px;
    margin-bottom: 1em;
    margin-left: 4em;
    border: 2px solid red;
    background: #e2f3f5;
  }
Ⓒ body {
    margin: 0 20%;
    border: 3px solid red;
    background-color: #e2f3f5;
  }
```

Margins

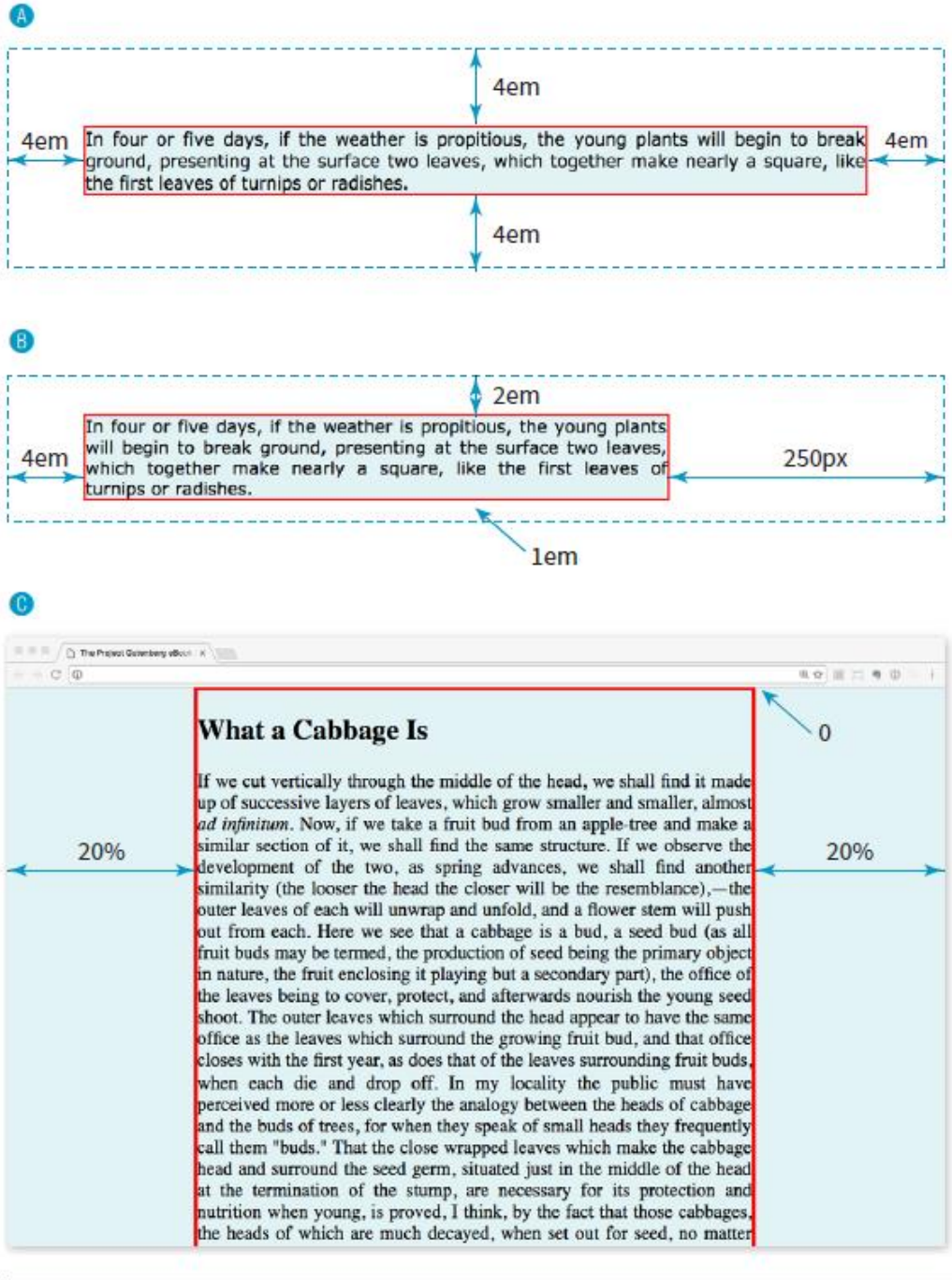


FIGURE 14-16. Applying margins to the **body** and to individual elements.

Margins

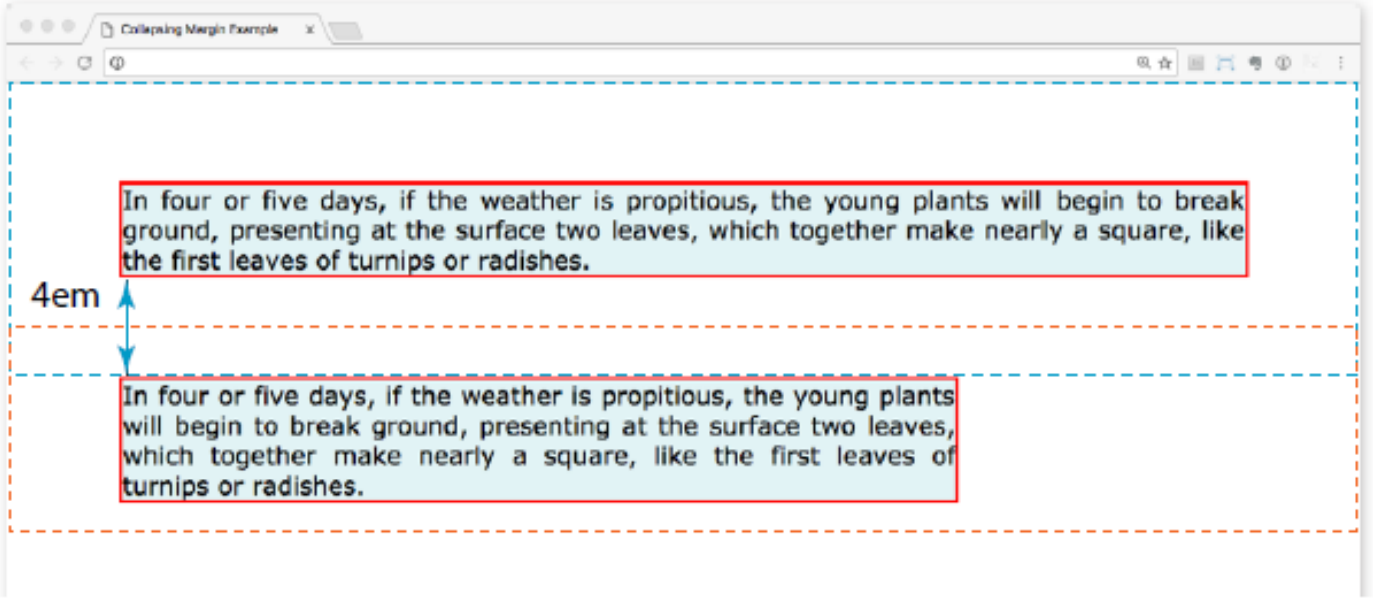


FIGURE 14-17. Vertical margins of neighboring elements collapse so that only the larger value is used.

Margin Behavior

Collapsing margins

```
em { margin: 2em; }
```

In four or five days, if the weather is propitious, the young plants will begin to *break ground*, presenting at the surface two leaves, which together make nearly a square, like the first leaves of turnips or radishes.

```
img { margin: 2em; }
```

In four or five days, if the weather is propitious, the young

plants will begin to break ground, presenting at the surface two leaves, which together make nearly a square, like the first leaves of turnips or radishes.

FIGURE 14-18. Examples of margins on inline elements. Only horizontal margins are rendered on non-replaced elements (top). Margins are rendered on all sides of replaced elements such as images.

Margin Behavior

Margins on inline elements

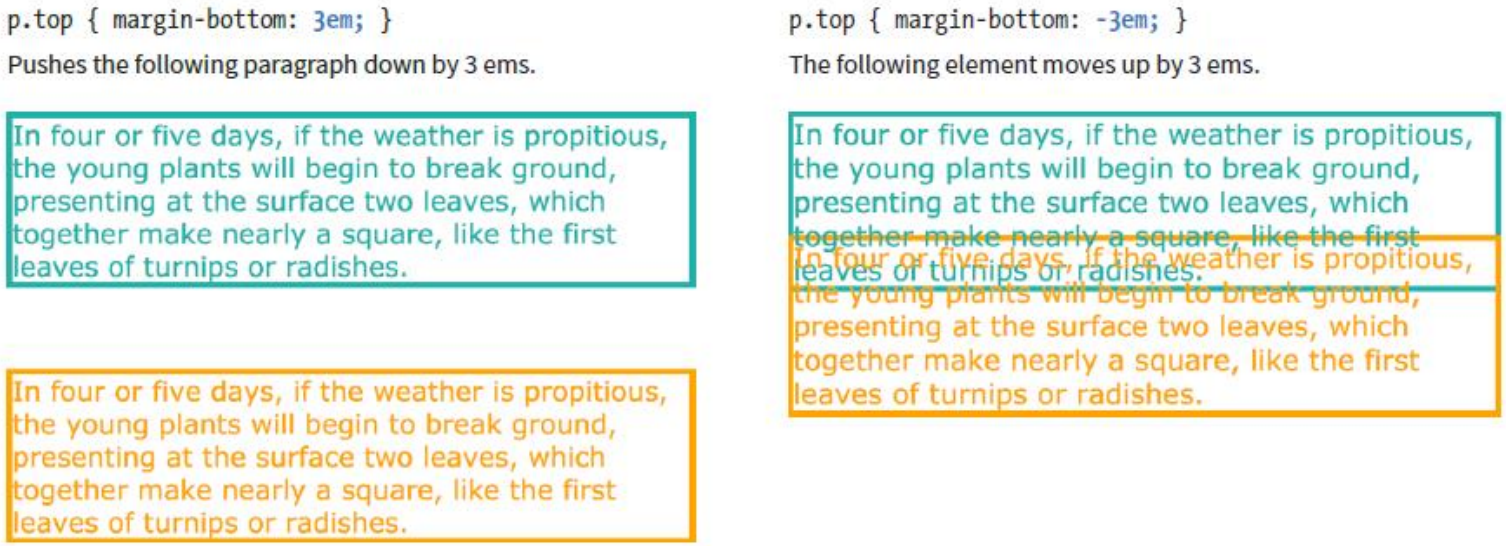


FIGURE 14-19. Using negative margins.

Margin Behavior

Negative margins

Display Styles

SECTION 5

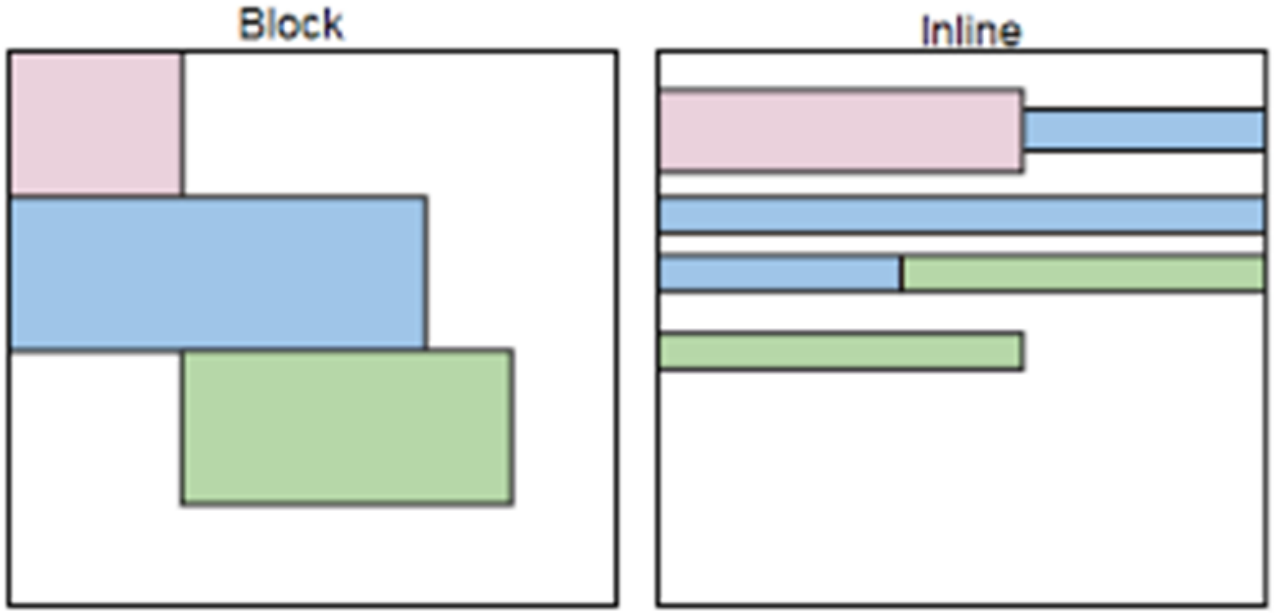
BLOCK-LEVEL ELEMENTS

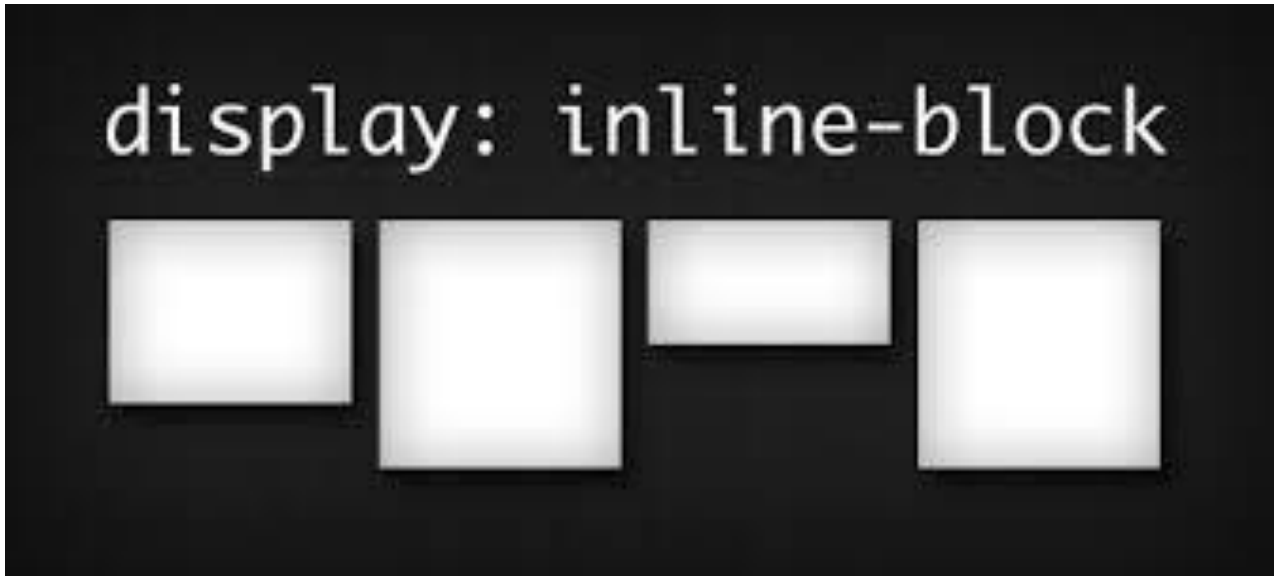
<address>	<article>	<aside>	<blockquote>	<canvas>	<dd>	
<dl>	<dt>	<fieldset>	<figcaption>	<figure>	<footer>	<form>
<h1>-<h6>	<header>	<hr>		<main>	<nav>	<noscript>
	<p>	<pre>	<section>	<table>	<tfoot>	
<video>						

INLINE ELEMENTS

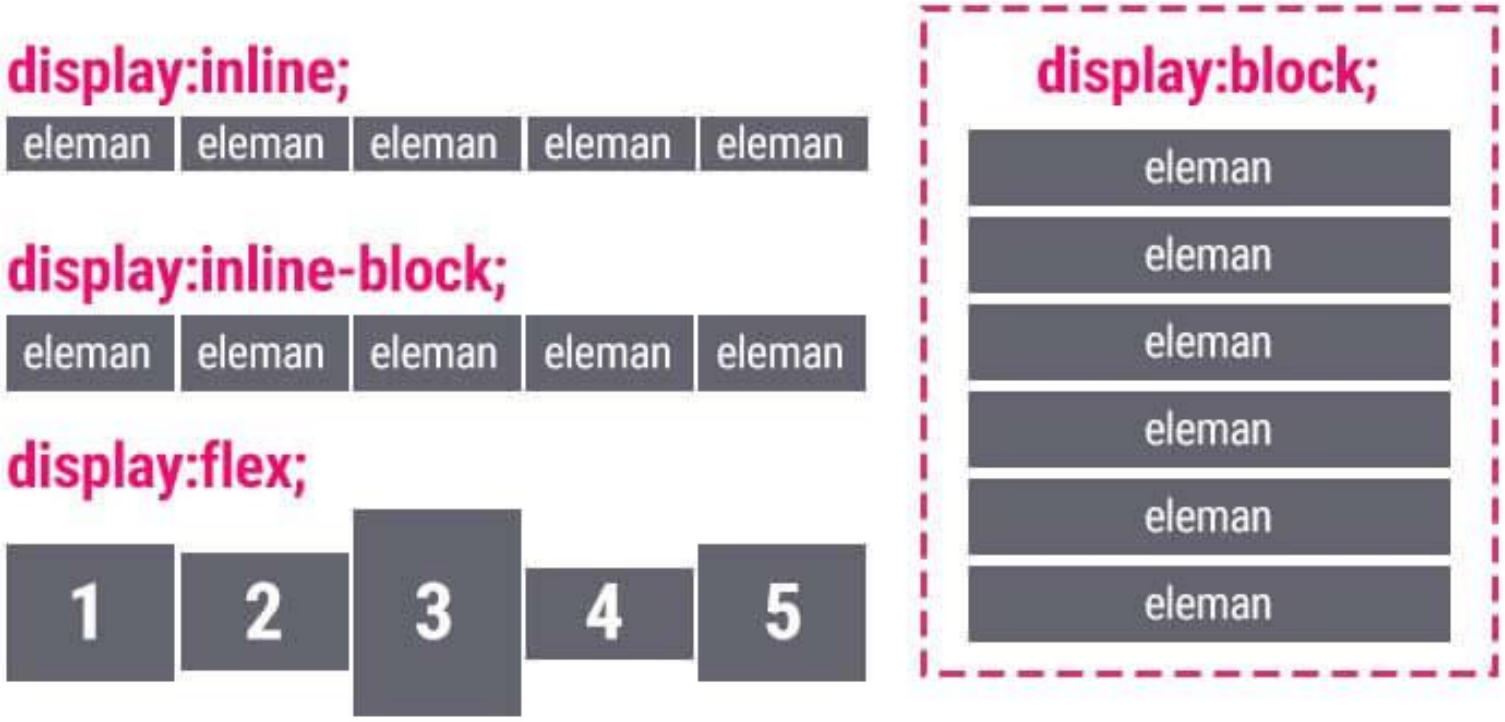
<a>	<abbr>	<acronym>	<b>	<bdo>	<big>	
<button>	<cite>	<code>	<dfn>		<i>	<img>
<input>	<kbd>	<label>	<map>	<object>	<output>	<q>
<samp>	<script>	<select>	<small>			<sub>
<sup>	<textarea>	<time>	<tt>	<var>		

display

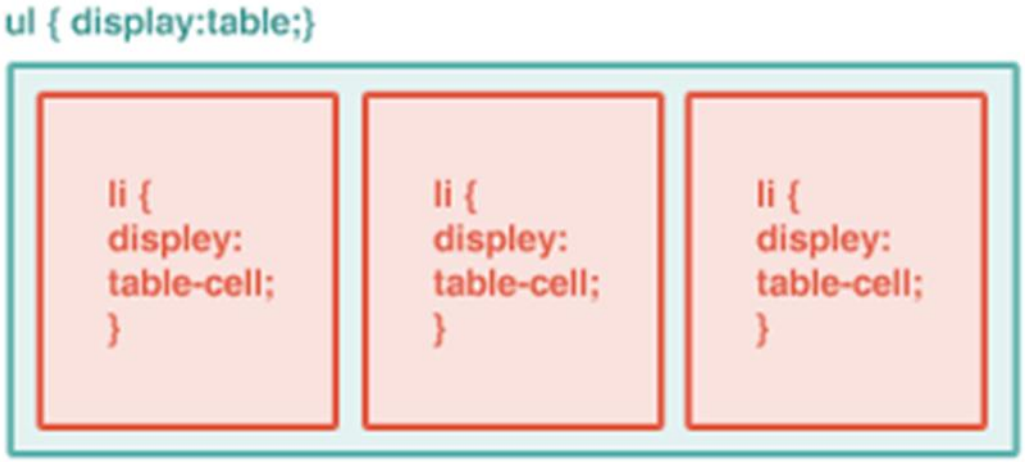




display


display:inline;
eleman eleman eleman eleman eleman
display:inline-block;
eleman eleman eleman eleman eleman
display:flex;
1 2 3 4 5
display:block;
eleman
eleman
eleman
eleman
eleman
eleman

display: table;



- Treat Elements like there are in a **table (like html)**: table-row-group, table-column-group, table cell, table-caption, table-row, table-footer-group, table-header-group, table-column, none,
- **line** is like ul or ol in html, **inline** is like li in html, **block** is like a bigger group, **inline-block** is like a row. **display** is used to specify the relative location of an element to other elements. **block** is a bigger area unit. It will not be pushed into a line when it does not fit.
- **display** property is only topological not logical.

Block Drop Shadows

- We’ve arrived at the last stop on the element box tour. In Chapter 12,
- Formatting Text, you learned about the **text-shadow** property, which adds a drop shadow to text. The **box-shadow** property applies a drop shadow around the entire visible element box (excluding the margin).

box-shadow

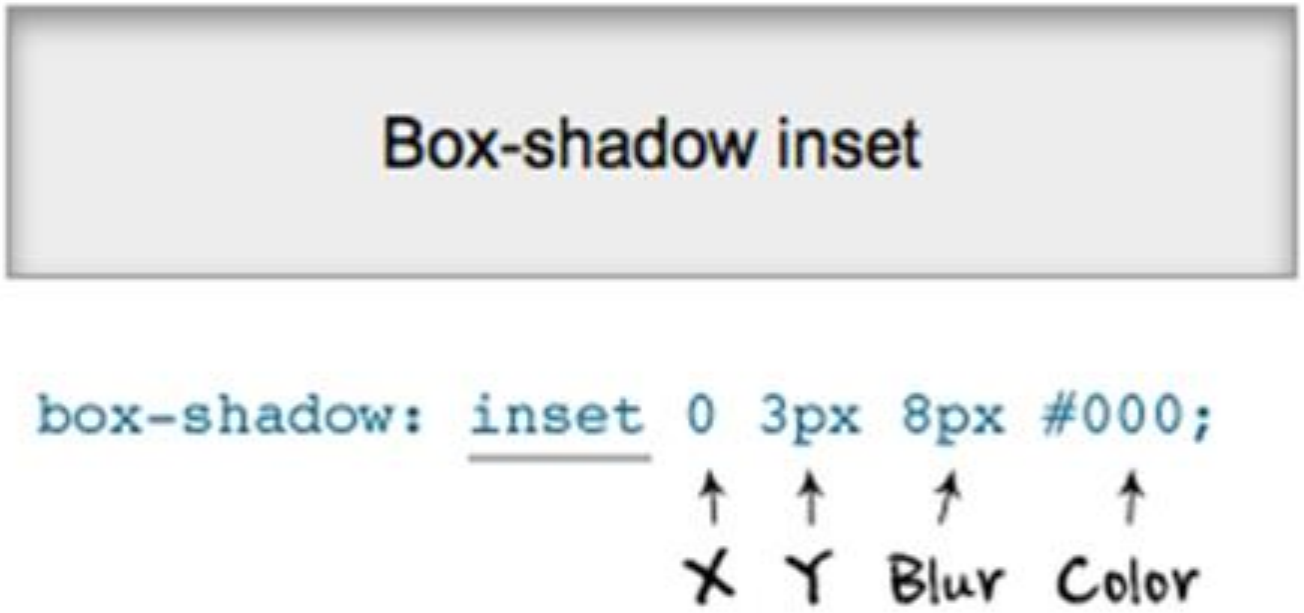
Values: *‘horizontal offset’ ‘vertical offset’ ‘blur distance’ ‘spread distance’ color*
inset | none

Default: none

Applies to: all elements

Inherits: no

Box Shadow



Box Shadow

box-shadow: x-offset y-offset **blur** **spread** color inset | none

```
box-shadow: inset 6px 6px 5px gray;
```

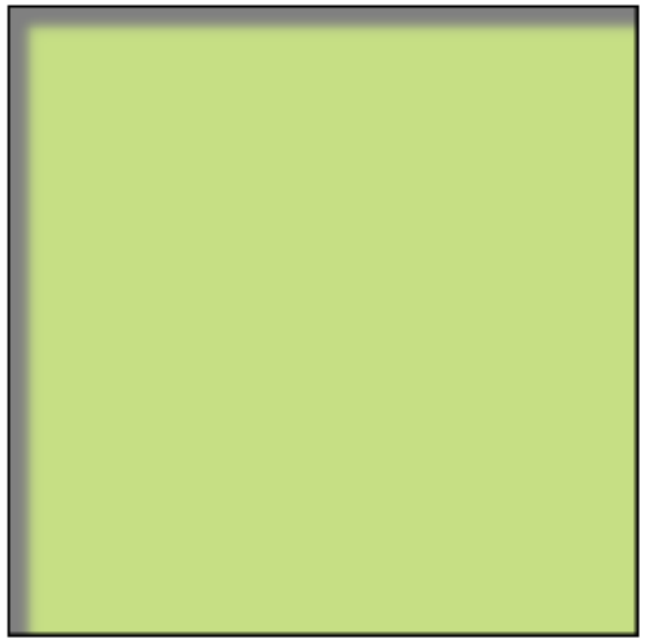


FIGURE 14-22. An inset box shadow renders on the inside of the element box.

Box Shadow

```
A box-shadow: 6px 6px gray;
B box-shadow: 6px 6px 5px gray; /* 5 pixel blur */
C box-shadow: 6px 6px 5px 10px gray; /* 5px blur, 10px spread */
```

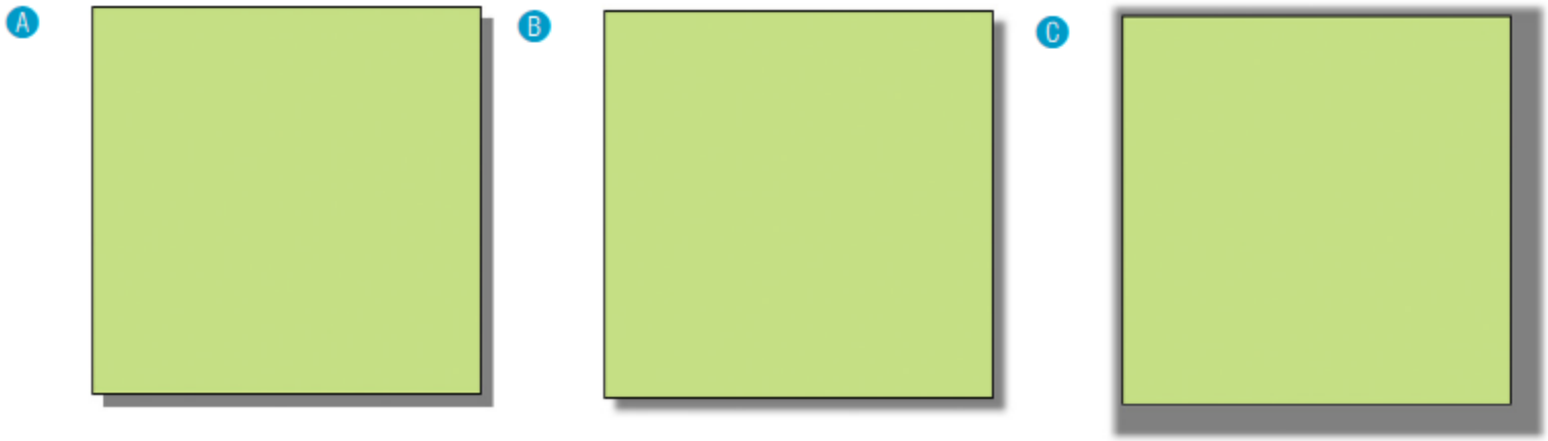
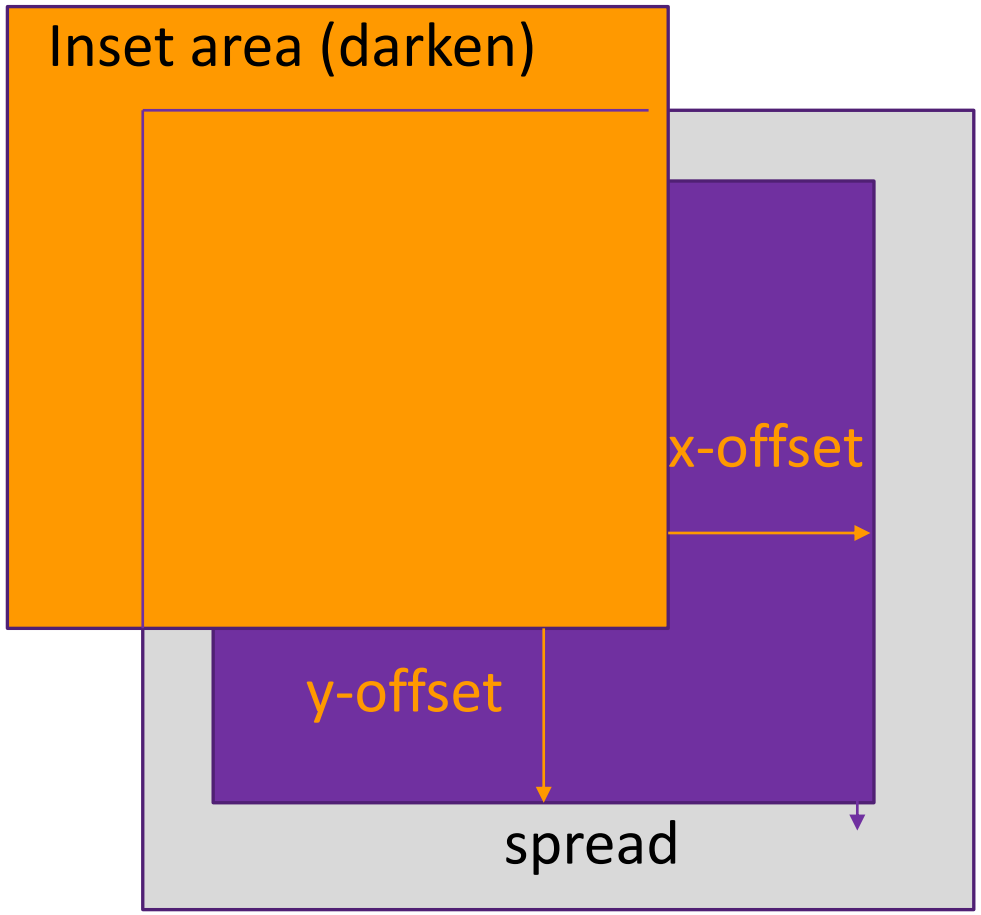
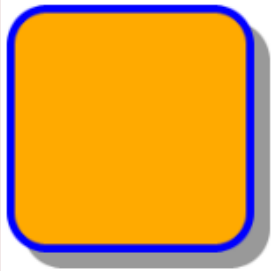
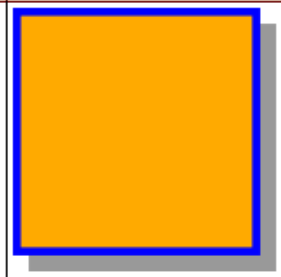
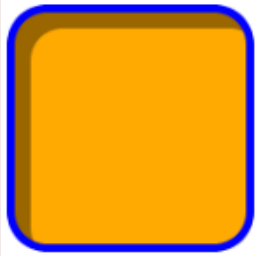
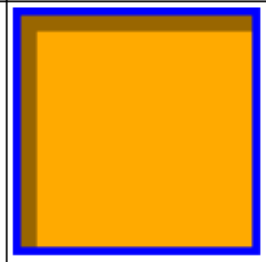
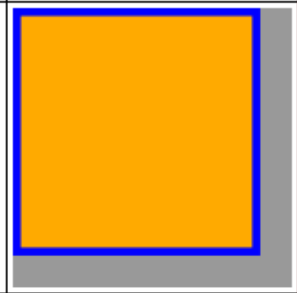
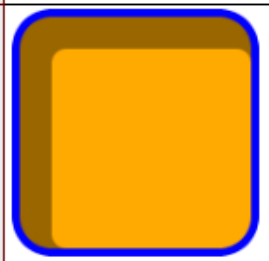
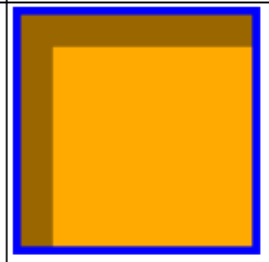


FIGURE 14-21. Adding drop shadows around an element with the **box-shadow** property.

Box Shadow

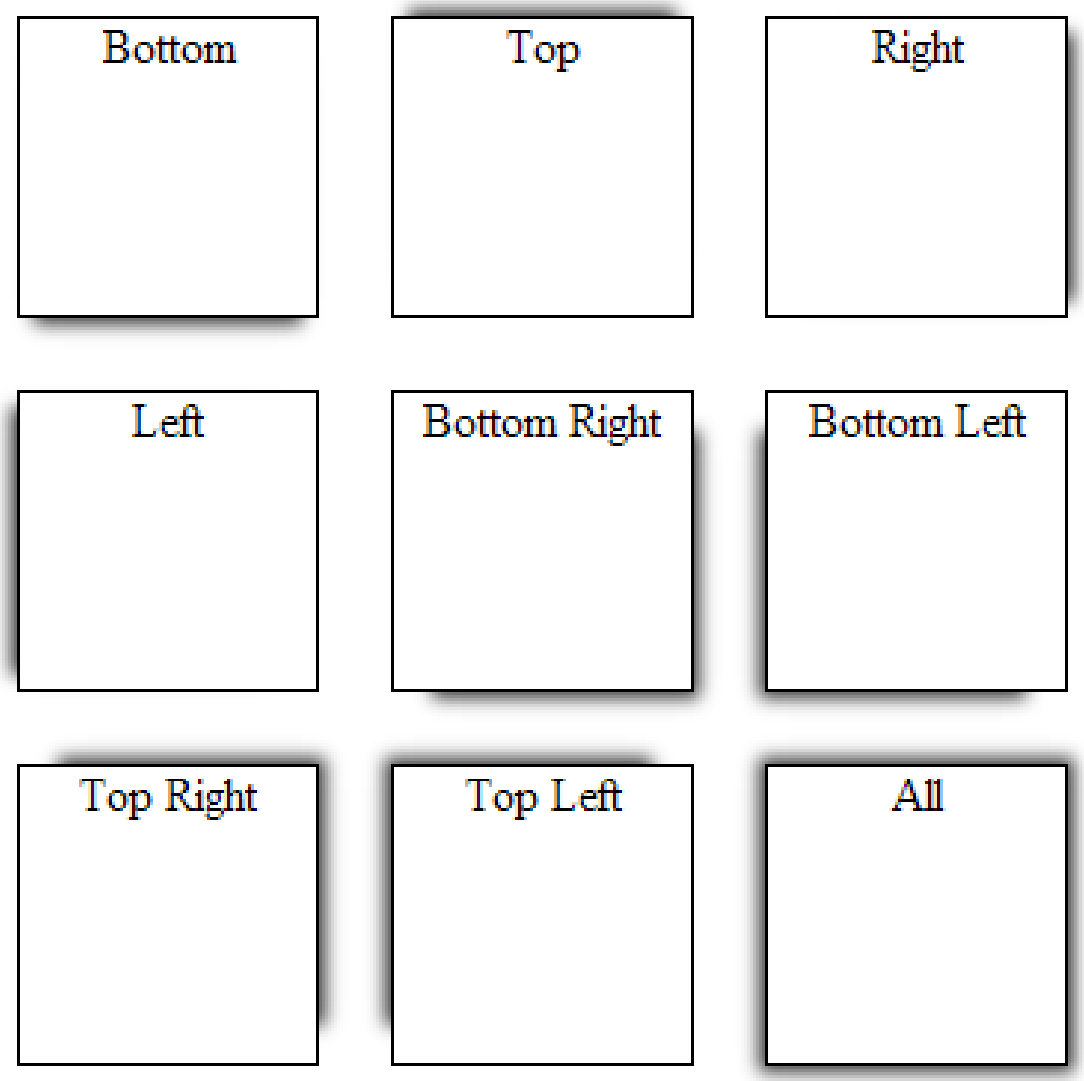


radius and shadow

border:5px solid blue; background-color:orange; width: 144px; height: 144px;	border-radius: 20px;	border-radius: 0;
box-shadow: rgba(0,0,0,0.4) 10px 10px;		
box-shadow: rgba(0,0,0,0.4) 10px 10px inset		
box-shadow: rgba(0,0,0,0.4) 10px 10px 0 10px /* spread */		
box-shadow: rgba(0,0,0,0.4) 10px 10px 0 10px /* spread */ inset		

radius and shadow

```
.shadow_bottom {
   box-shadow: 0px 8px 8px -6px #333; }
.shadow_top {
   box-shadow: 0px -8px 8px -6px #333; }
.shadow_right {
   box-shadow: 8px 0px 8px -6px #333; }
.shadow_left {
   box-shadow: -8px 0px 8px -6px #333; }
.shadow_bottomright {
   box-shadow: 8px 8px 8px -6px #333; }
.shadow_bottomleft {
   box-shadow: -8px 8px 8px -6px #333; }
.shadow_topright {
   box-shadow: 8px -8px 8px -6px #333; }
.shadow_topleft {
   box-shadow: -8px -8px 8px -6px #333; }
.shadow_all {
   box-shadow: 0px 0px 8px 2px #333; }
```

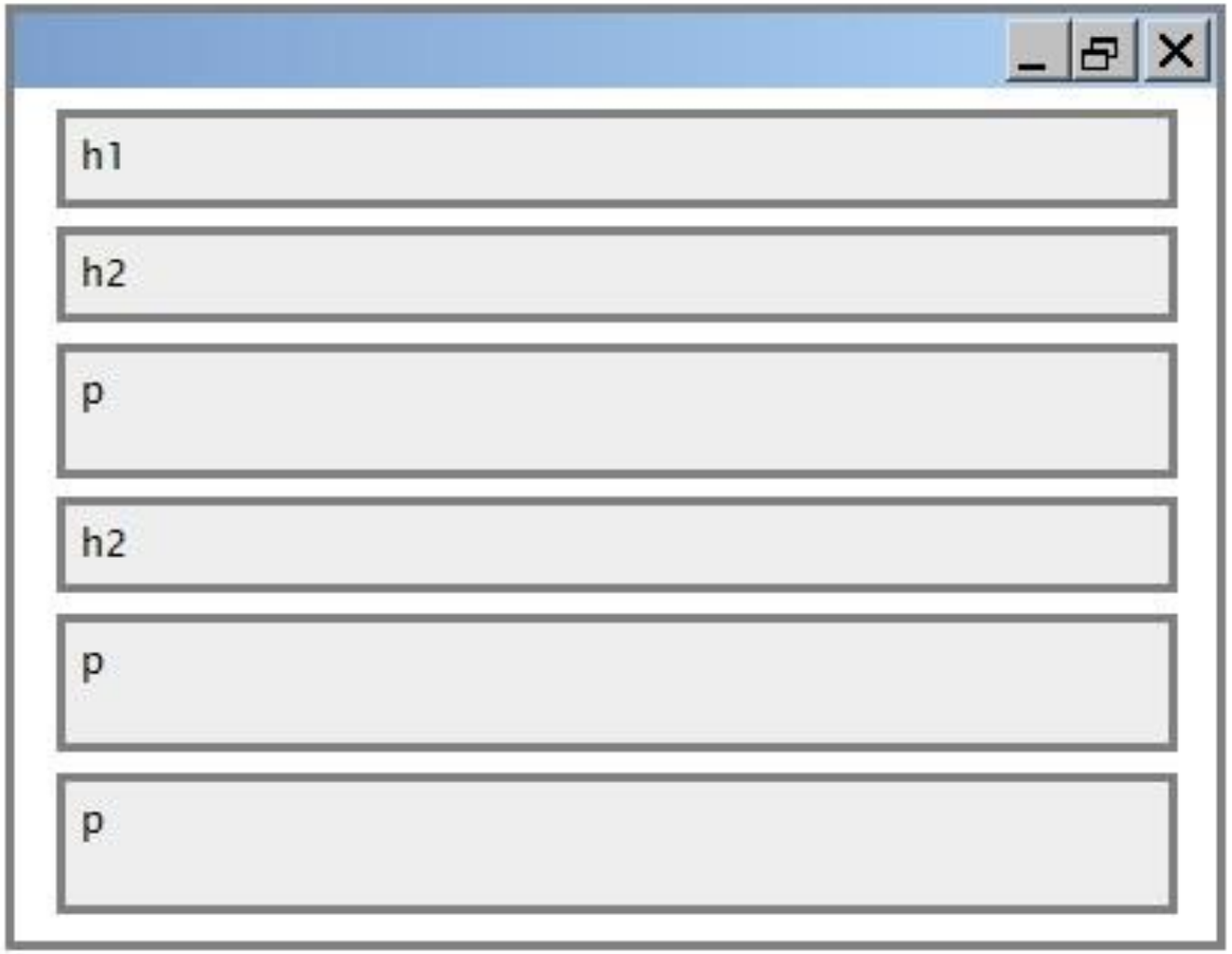


Flow

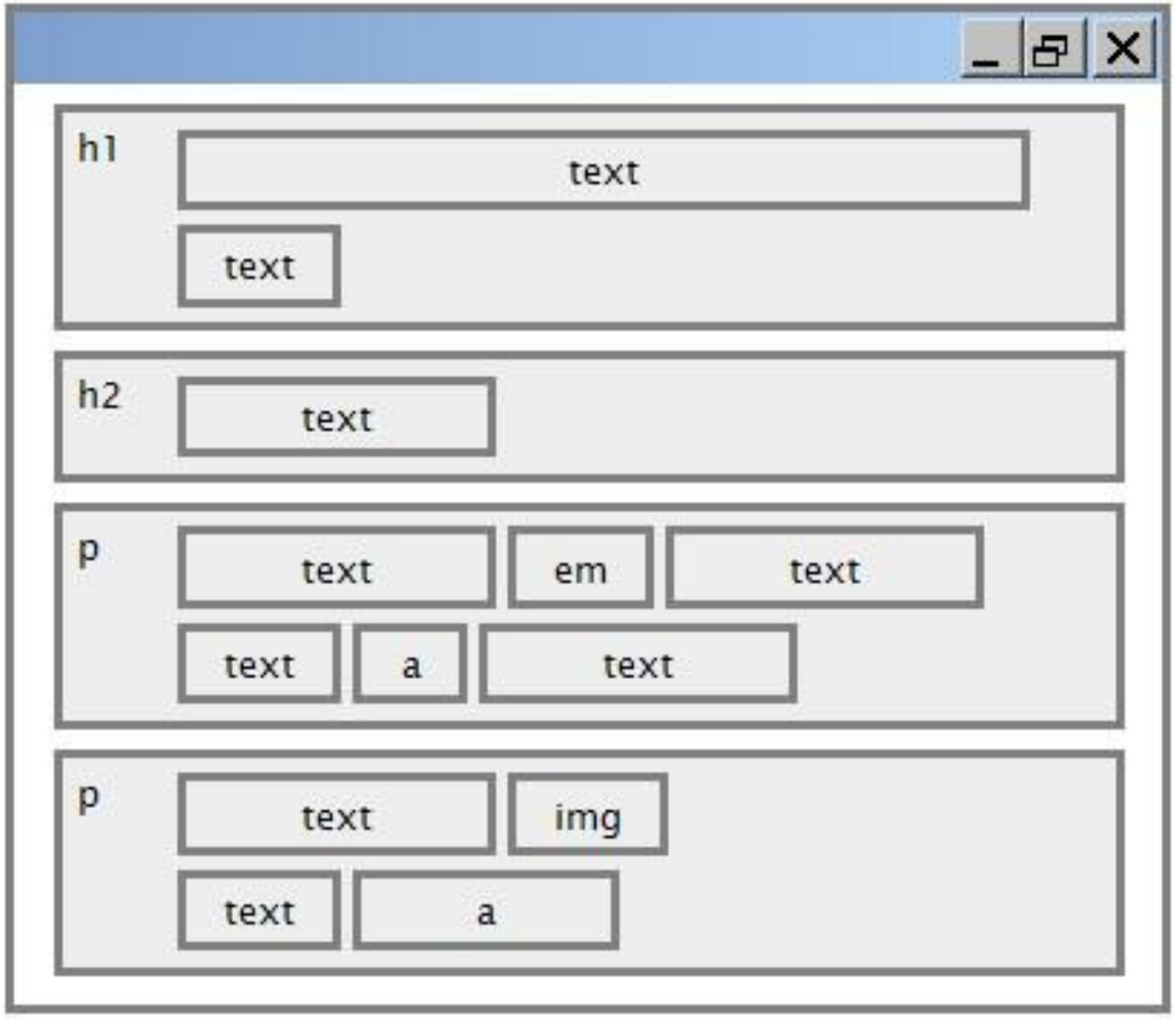
SECTION 6

Normal Flow

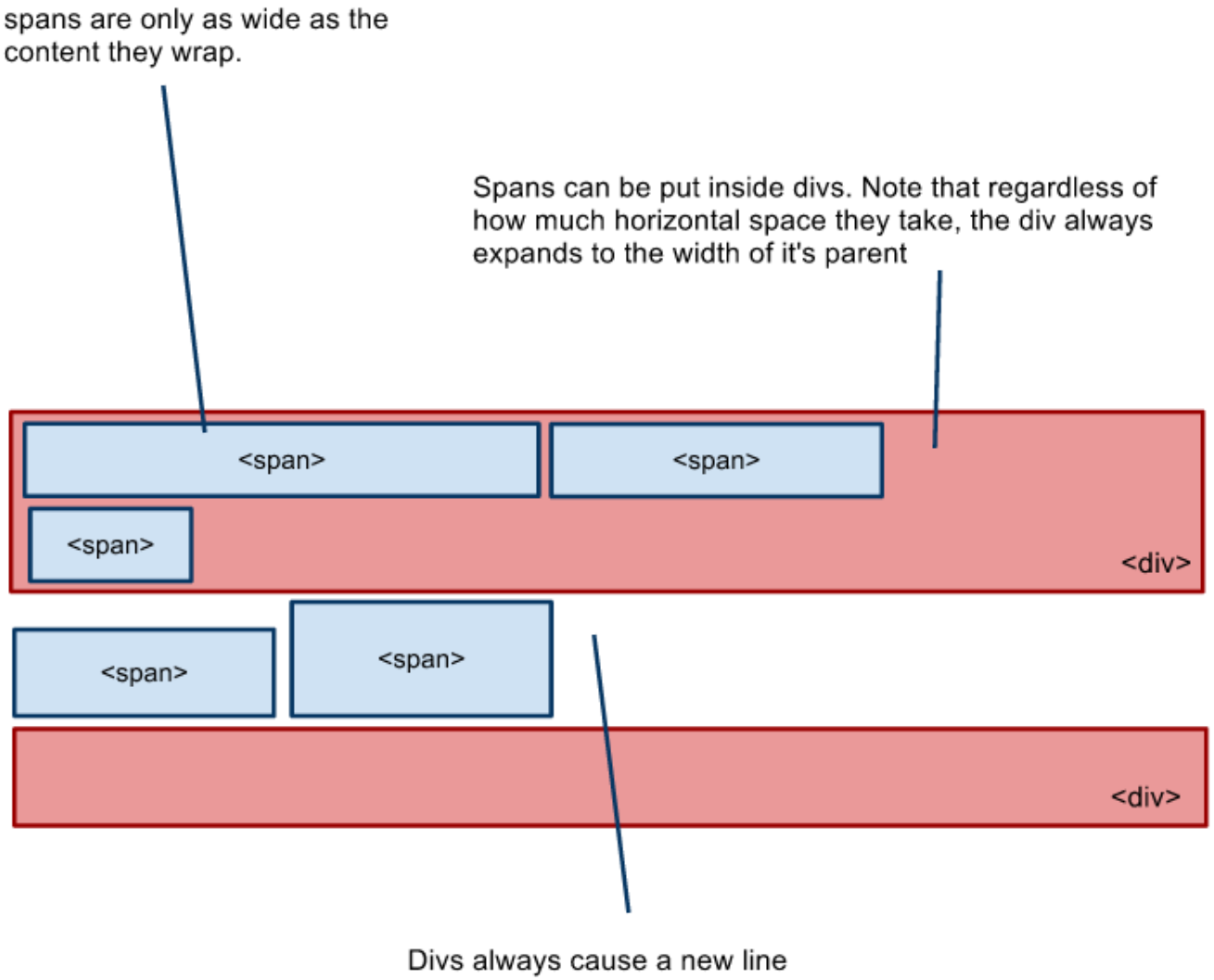
- In the CSS layout model, text elements are laid out from top to bottom, in the order in which they appear in the source, and from left to right (in the left to right reading languages)
- **Block** elements stack up on the top of one another and fill the available width of the browser window or other containing element.
- **Inline** elements and text characters line up next to one another to fill the block elements.
- When the window or containing element is resized, the block elements expand or contract to the new width, and the inline content reflows to fit.



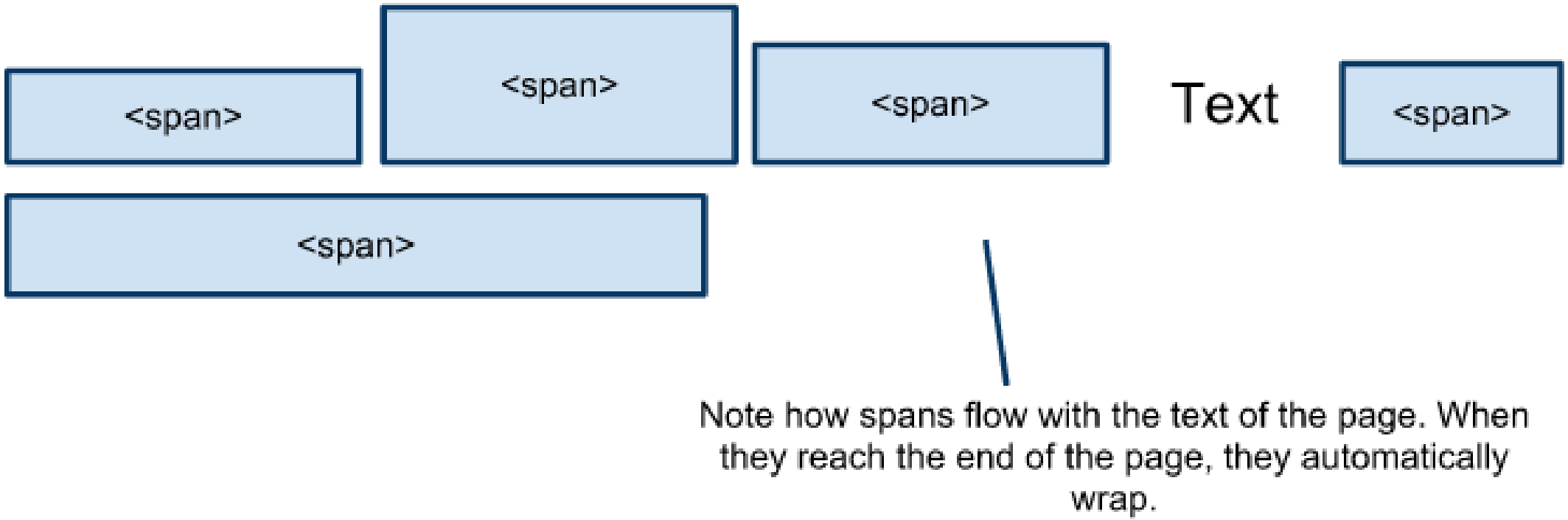
Normal Flow



Document Flow: Block and Inline Elements



always start a new line



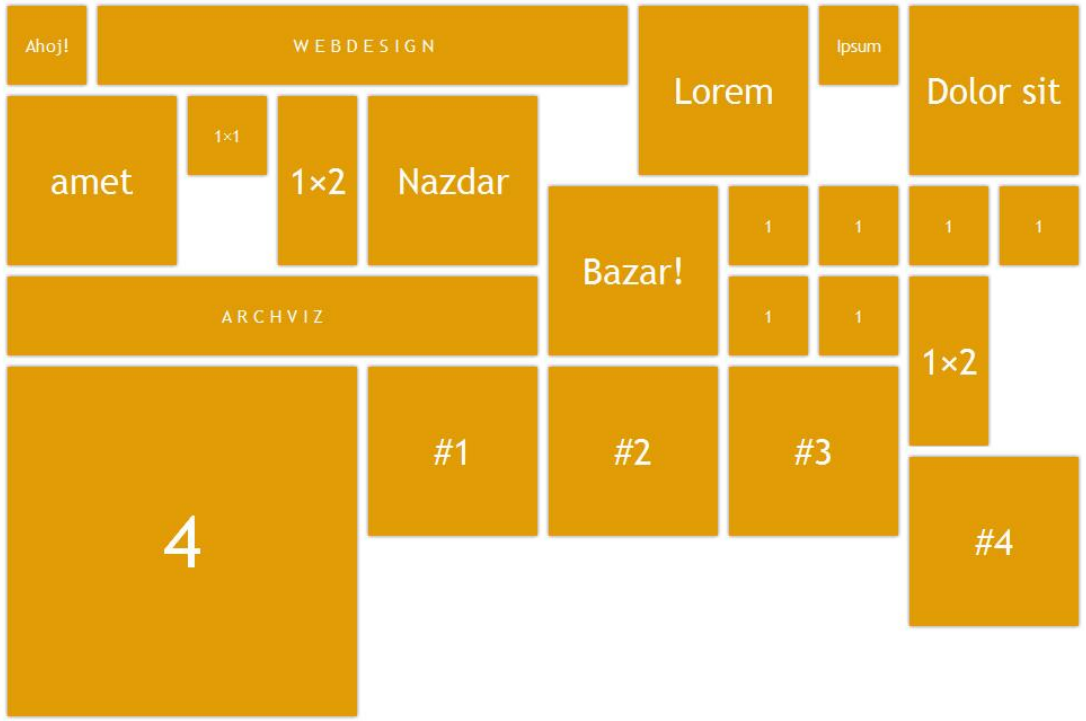
automatically wrap

Floating

SECTION 7

Floating

move an element as far as possible to the left or right



Floating

move an element as far as possible to the left or right

Floating

- Simply stated, the **float** property moves an element as far as possible to the left or right, allowing the following content to wrap around it. It is a unique feature built into CSS with some interesting behaviors.

float

Values: left | right | none
Default: none
Applies to: all elements
Inherits: no

THE MARKUP

```
<p><img src="icecreambowl.png" alt=""> After the cream is frozen rather stiff,…
```

THE STYLES

```
img {
    float: right;
}
```

Inline image in the normal flow

Space next to image is held clear

After the cream is frozen rather stiff, prepare a tub or bucket of coarsely chopped ice, with one-half less salt than you use for freezing. To each ten pounds of ice allow one quart of rock salt. Sprinkle a little rock salt in the bottom of your bucket or tub, then put over a layer of cracked ice, another layer of salt and cracked ice, and on this stand your mold, which is not filled, but is covered with a lid, and pack it all around, leaving the top, of course, to pack later on. Take your freezer near this tub. Remove the lid from the mold, and pack in the cream, smoothing it down until you have filled it to overflowing. Smooth the top with a spatula or limber knife, put over a sheet of waxed paper and adjust the lid.

Inline image floated to the right

Image moves over, and text wraps around it

After the cream is frozen rather stiff, prepare a tub or bucket of coarsely chopped ice, with one-half less salt than you use for freezing. To each ten pounds of ice allow one quart of rock salt. Sprinkle a little rock salt in the bottom of your bucket or tub, then put over a layer of cracked ice, another layer of salt and cracked ice, and on this stand your mold, which is not filled, but is covered with a lid, and pack it all around, leaving the top, of course, to pack later on. Take your freezer near this tub. Remove the lid from the mold, and pack in the cream, smoothing it down until you have filled it to overflowing. Smooth the top with a spatula or limber knife, put over a sheet of waxed paper and adjust the lid.

FIGURE 15-2. The layout of an image in the normal flow (top), and with the `float` property applied (bottom).

Floating

```
img {
  float: right;
  margin: 1em;
}
```

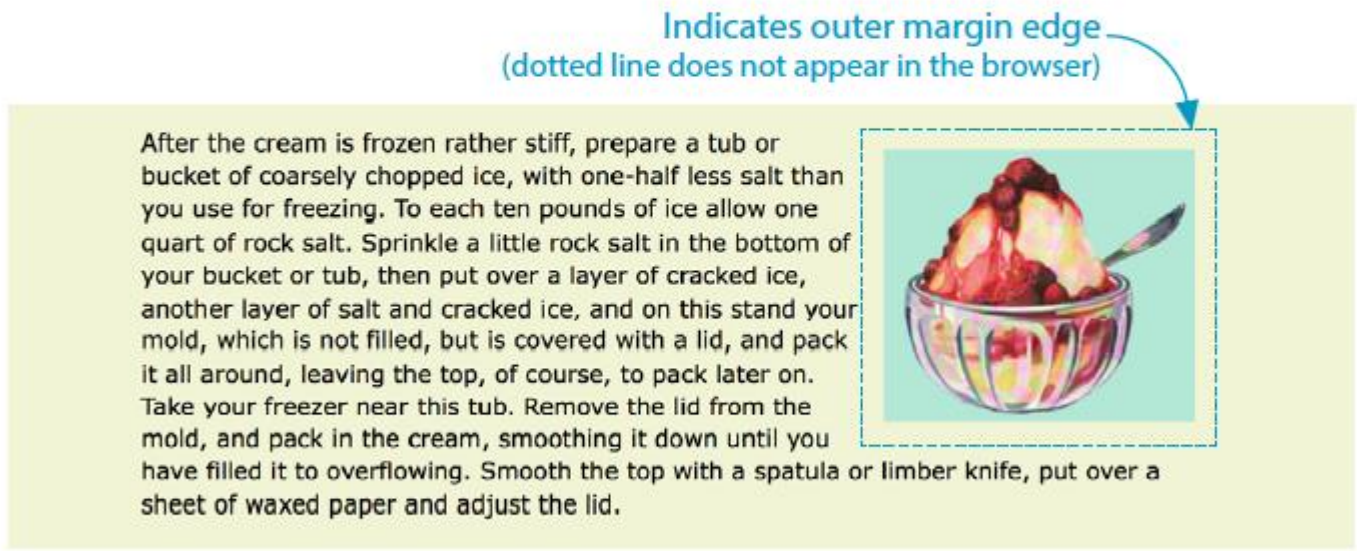
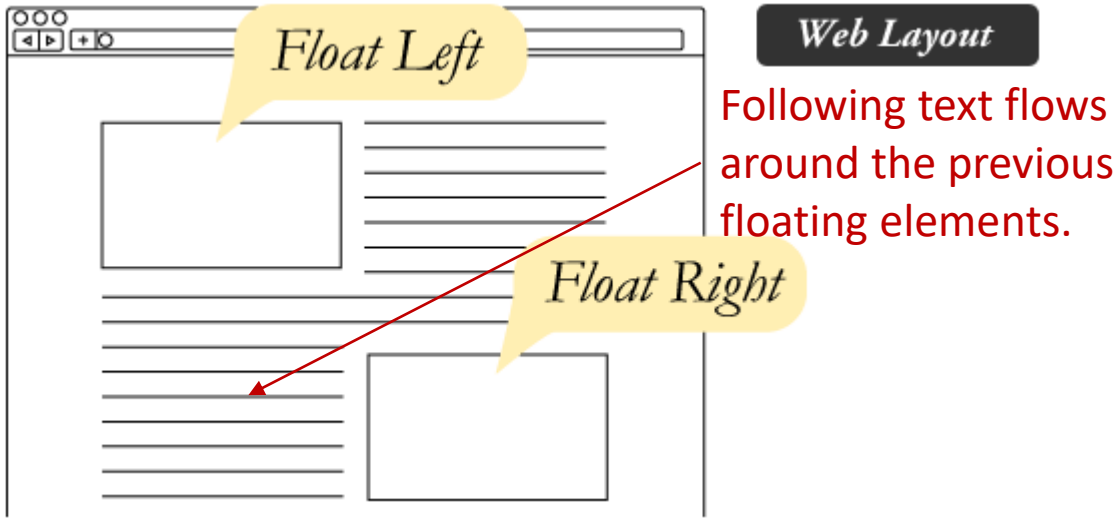


FIGURE 15-3. Adding a 1em margin around the floated image.

Floating

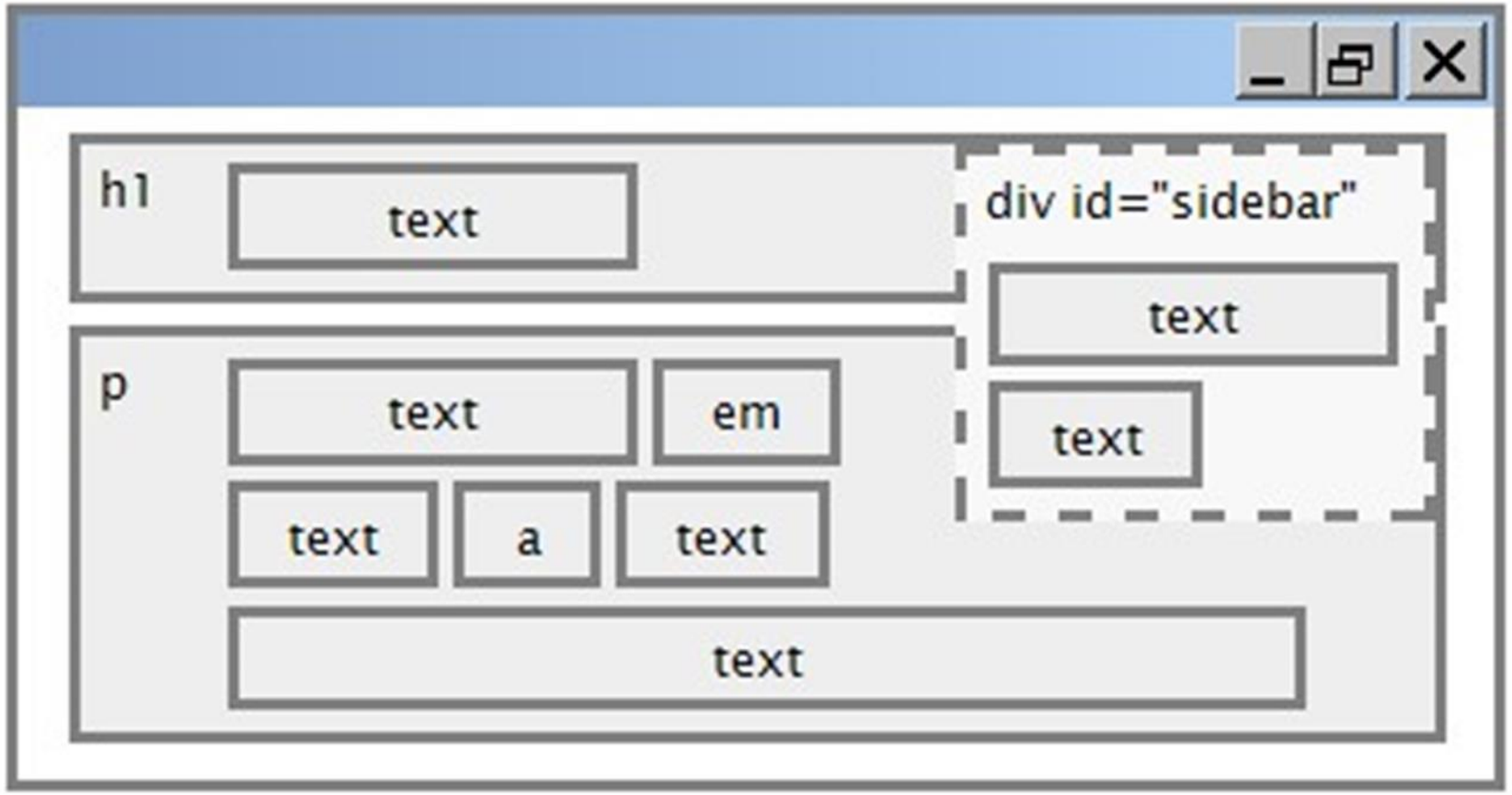
- a ***floating*** element is removed from normal document flow
- underlying text wraps around it as necessary



property	description
`float`	side to hover on; can be `left`, `right`, or `none` (default)

Floating

float: left | right |none | inherit



Floating Out of Normal Flow

Design of the Floated Elements

A floated element is like an island in a stream.

First and foremost, you can see that the image is removed from its position in the normal flow yet continues to influence the surrounding content. The subsequent paragraph text reflows to make room for the floated **img** element. One popular analogy compares floats to islands in a stream—they are not in the flow, but the stream has to flow around them. This behavior is unique to floated elements.

Floats stay in the content area of the containing element.

It is also important to note that the floated image is placed within the *content area* (the inner edges) of the paragraph that contains it. It does not extend into the padding area of the paragraph.

Margins are maintained.

In addition, margins are held on all sides of the floated image, as indicated in FIGURE 15-3 by the dotted line. In other words, the entire element box, from outer edge to outer edge, is floated.

none 1 before

none 2 before

left #1 left #2 none 1 after right #2 right #1

none 2 after

- using Firebug, toggle the above `divs` from being aligned to floated...

I am not floating, no width set

I am floating right, no width set

I am floating right, no width set, but my text is very long so this paragraph doesn't really seem like it's floating at all, darn

I am not floating, 45% width

I am floating right, 45% width

- often floating elements should have a `width` property value
  - if no `width` is specified, other content may be unable to wrap around the floating element

Floating inline and block elements

Floating Inline and Block elements

Floating an inline text element

THE MARKUP

```
<p><span class="tip">TIP: Make sure that your packing tub or bucket has a hole below the top of the mold so the water will drain off.</span>After the cream is frozen rather stiff, prepare a tub or bucket of…
</p>
```

THE STYLES

```
span.tip {
   float: right;
   margin: 1em;
   width: 200px;
   color: #fff;
   background-color: lightseagreen;
   padding: 1em;
}
```

Floating Inline and Block elements

Floating an inline text element

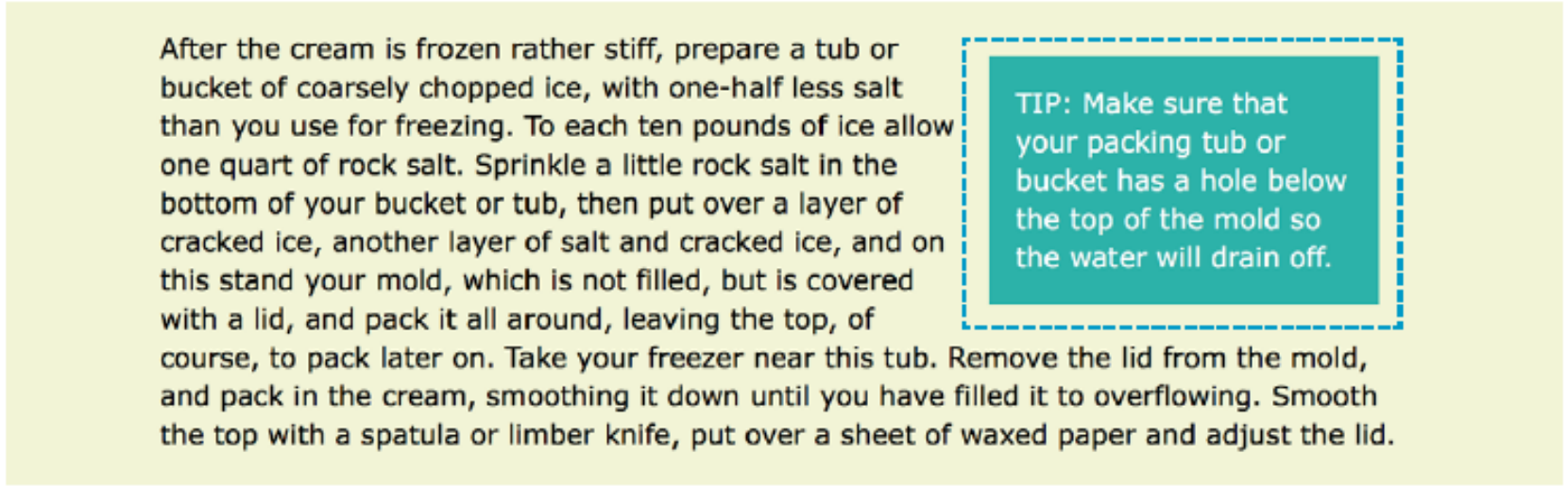


FIGURE 15-4. Floating an inline text (non-replaced) element.

Floating Inline and Block elements

Floating an inline text element

There are some subtle things at work here that bear pointing out:

1. Always provide a width for floated text elements.
2. Floated inline elements behave as block elements.
3. Margins on floated elements do not collapse.

Floating Block Elements

THE MARKUP

```
<p>If you wish to pack ice cream...</p>
<p id="float">After the ice cream is rather
stiff, ... </p>
<p>Make sure that your packing tub or bucket
... </p>
<p>As cold water is warmer than the ordinary
... </p>
```

THE STYLES

```
p {
    border: 2px red solid;
    }
    #float {
    float: left;
    width: 300px;
    margin: 1em;
    background: white;
}
```

Floating Block Elements

If you wish to pack ice cream and serve it in forms or shapes, it must be molded after the freezing. The handiest of all of these molds is either the brick or the melon mold.

After the cream is frozen rather stiff, prepare a tub or bucket of coarsely chopped ice, with one-half less salt than you use for freezing. To each ten pounds of ice allow one quart of rock salt. Sprinkle a little rock salt in the bottom of your bucket or tub, then put over a layer of cracked ice, another layer of salt and cracked ice, and on this stand your mold, which is not filled, but is covered with a lid, and pack it all around, leaving the top, of course, to pack later on. Take your freezer near this tub.

Remove the lid from the mold, and pack in the cream, smoothing it down until you have filled it to overflowing. Smooth the top with a spatula or limber knife, put over a sheet of waxed paper and adjust the lid. Have a strip of muslin or cheese cloth dipped in hot paraffin or suet and quickly bind the seam of the lid. This will remove all danger of salt water entering the pudding. Now cover the mold thoroughly with ice and salt.

Make sure that your packing tub or bucket has a hole below the top of the mold, so that the salt water will be drained off. If you are packing in small molds, each mold, as fast as it is closed, should be wrapped in wax paper and put down into the salt and ice. These must be filled quickly and packed.

As cold water is warmer than the ordinary freezing mixture, after you lift the can or mold, wipe off the salt, hold it for a minute under the cold water spigot, then quickly wipe the top and bottom and remove the lid. Loosen the pudding with a limber knife, hold the mold a little slanting, give it a shake, and nine times out of ten it will come out quickly, having the perfect shape of the can or mold. If the cream still sticks and refuses to come out, wipe the mold with a towel wrung from warm water. Hot water spoils the gloss of puddings, and unless you know exactly how to use it, the cream is too much melted to garnish.

If you wish to pack ice cream and serve it in forms or shapes, it must be molded after the freezing. The handiest of all of these molds is either the brick or the melon mold.

After the cream is frozen rather stiff, prepare a tub or bucket of coarsely chopped ice, with one-half less salt than you use for freezing. To each ten pounds of ice allow one quart of rock salt. Sprinkle a little rock salt in the bottom of your bucket or tub, then put over a layer of cracked ice, another layer of salt and cracked ice, and on this stand your mold, which is not filled, but is covered with a lid, and pack it all around, leaving the top, of course, to pack later on. Take your freezer near this tub.

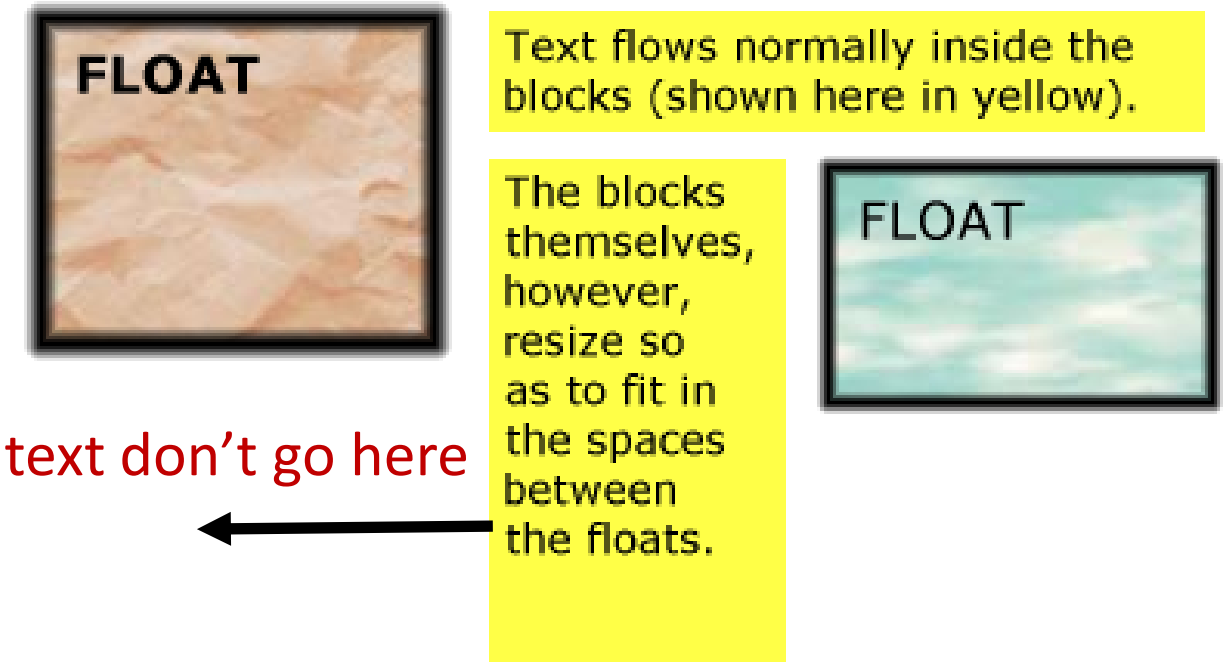
Remove the lid from the mold, and pack in the cream, smoothing it down until you have filled it to overflowing. Smooth the top with a spatula or limber knife, put over a sheet of waxed paper and adjust the lid. Have a strip of muslin or cheese cloth dipped in hot paraffin or suet and quickly bind the seam of the lid. This will remove all danger of salt water entering the pudding. Now cover the mold thoroughly with ice and salt.

Make sure that your packing tub or bucket has a hole below the top of the mold, so that the salt water will be drained off. If you are packing in small molds, each mold, as fast as it is closed, should be wrapped in wax paper and put down into the salt and ice. These must be filled quickly and packed.

As cold water is warmer than the ordinary freezing mixture, after you lift the can or mold, wipe off the salt, hold it for a minute under the cold water spigot, then quickly wipe the top and bottom and remove the lid. Loosen the pudding with a limber knife, hold the mold a little slanting, give it a shake, and nine times out of ten it will come out quickly, having the perfect shape of the can or mold. If the cream still sticks and refuses to come out, wipe the mold with a towel wrung from warm water. Hot water spoils the gloss of puddings, and unless you know exactly how to use it, the cream is too much melted to garnish.

FIGURE 15-5. Floating a block-level element.

Floating block Elements



Guidelines for Floating Block Elements

- You must provide a width for floated block elements.
- Elements do not float higher than their reference in the source.
- Non-floated elements maintain the normal flow.

clear

Values:	left | right | both | none
Default:	none
Applies to:	block-level elements only
Inherits:	no

Clearing Floated Elements

```
img {
  float: left;
  margin-right: .5em;
}
h2 {
  clear: left;
  margin-top: 2em;
}
```

If pure raw cream is stirred rapidly, it swells and becomes frothy, like the beaten whites of eggs, and is "whipped cream." To prevent this in making Philadelphia Ice Cream, one-half the cream is scalded, and when it is *very* cold, the remaining half of raw cream is added. This gives the smooth, light and rich consistency which makes these creams so different from others.

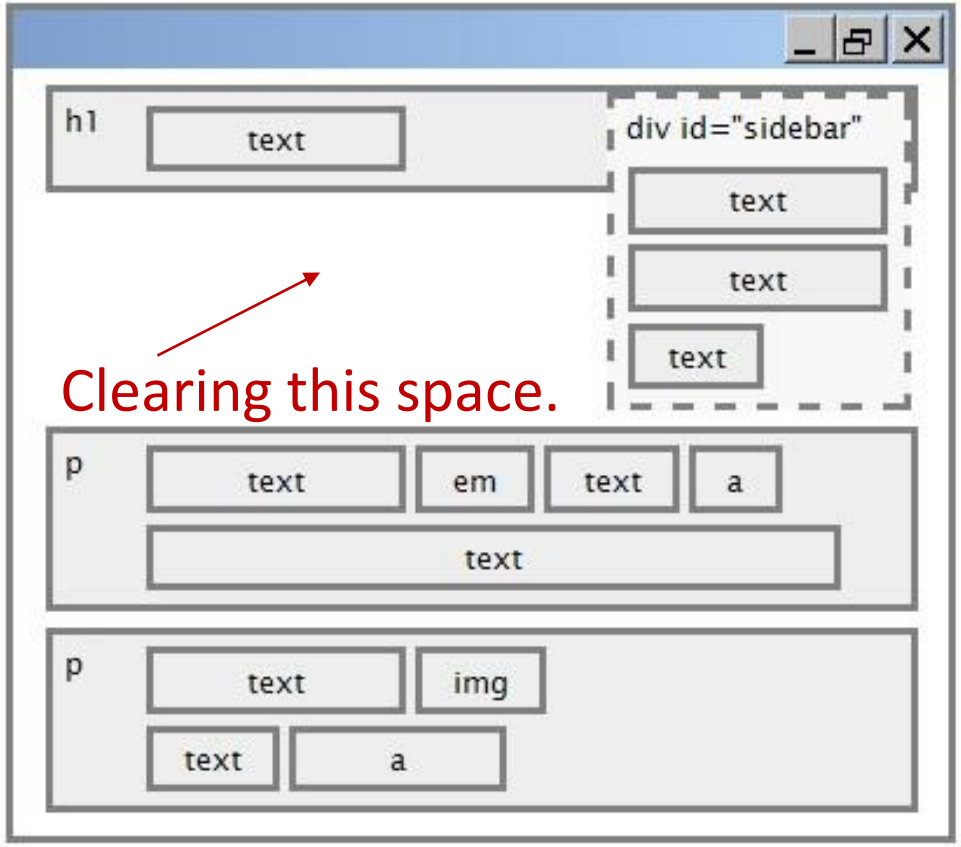
USE OF FRUITS

Use fresh fruits in the summer and the best canned unsweetened fruits in the winter. If sweetened fruits must be used, cut down the given quantity of sugar. Where acid fruits are used, they should be added to the cream after it is partly frozen.

The time for freezing varies according to the quality of cream or milk or water; water ices require a longer time than ice creams. It is not well to freeze the mixtures too rapidly; they are apt to be coarse, not smooth, and if they are churned before the mixture is icy cold they will be greasy or "buttery."

FIGURE 15-6. Clearing a left-floated element.

Clearing Floated Elements




```
div#sidebar { float: right; }
p { clear: right; }
```

property	description
`clear`	disallows floating elements from overlapping this element; can be `left`, `right`, `both`, or `none` (default)

Clearing Floated Elements

```html
<p><img src="images/homestar_runner.png" alt="homestar runner" />
  Homestar Runner is a Flash animated Internet cartoon.
  It mixes surreal humour with ....</p>
```
HTML

```css
p { border: 2px dashed black; }
img { float: right; }
```
CSS

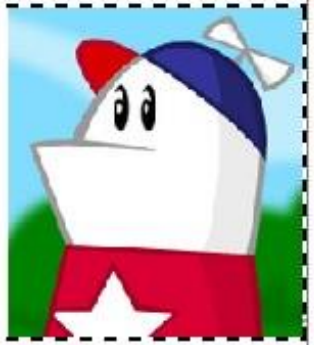
Homestar Runner is a Flash animated Internet cartoon. It mixes surreal humour with ....

- We want the p containing the image to extend downward so that its border encloses the entire image

Floating right example (overflow)

```css
p { border: 2px dashed black; overflow: hidden; }
```

Homestar Runner is a Flash animated Internet cartoon. It mixes surreal humour with ....



property	description
`overflow`	specifies what to do if an element's content is too large; can be `auto`, `visible`, `hidden`, or `scroll`

Overflow property

Floating Multiple Elements

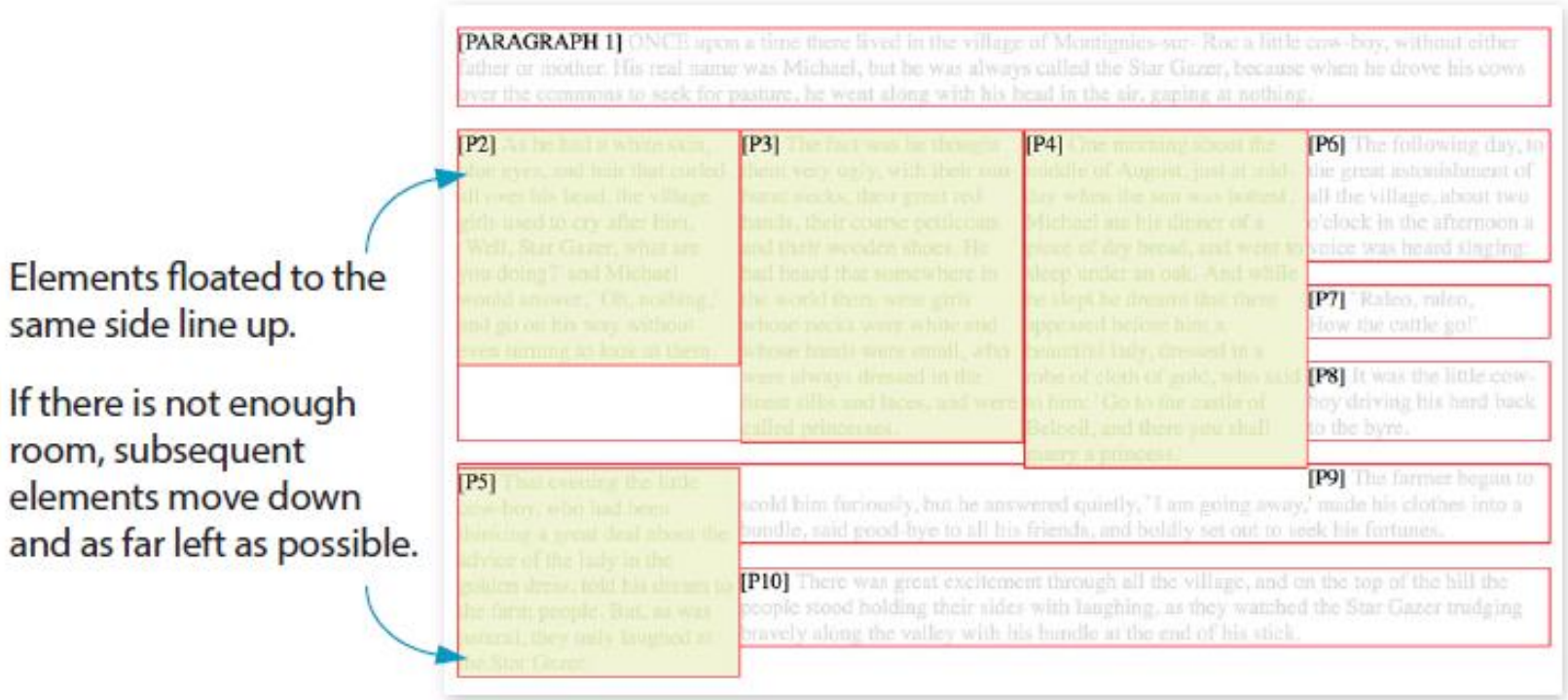


FIGURE 15-8. Multiple floated elements line up and do not overlap.

Floating Multiple Elements

THE MARKUP

```
<p>[PARAGRAPH 1] ONCE upon a time…</p>
<p class="float">[P2]…</p>
<p class="float">[P3]…</p>
<p class="float">[P4]…</p>
<p class="float">[P5]…</p>
<p>[P6]…</p>
<p>[P7]…</p>
<p>[P8]…</p>
<p>[P9]…</p>
<p>[P10]…</p>
```

THE STYLES

```
p.float {
    float: left;
    width: 200px;
    margin: 0px;
    background: #F2F5d5;
    color: #DAEAB1;
}
```

Containing Floats

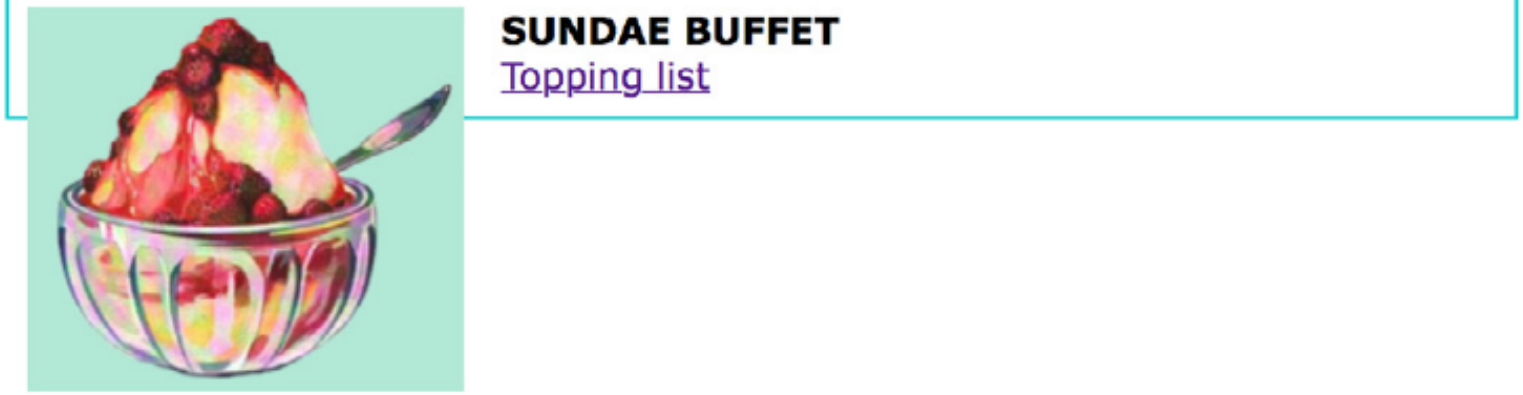


FIGURE 15-9. The containing element does not expand to accommodate the floated image as indicated by its blue border.

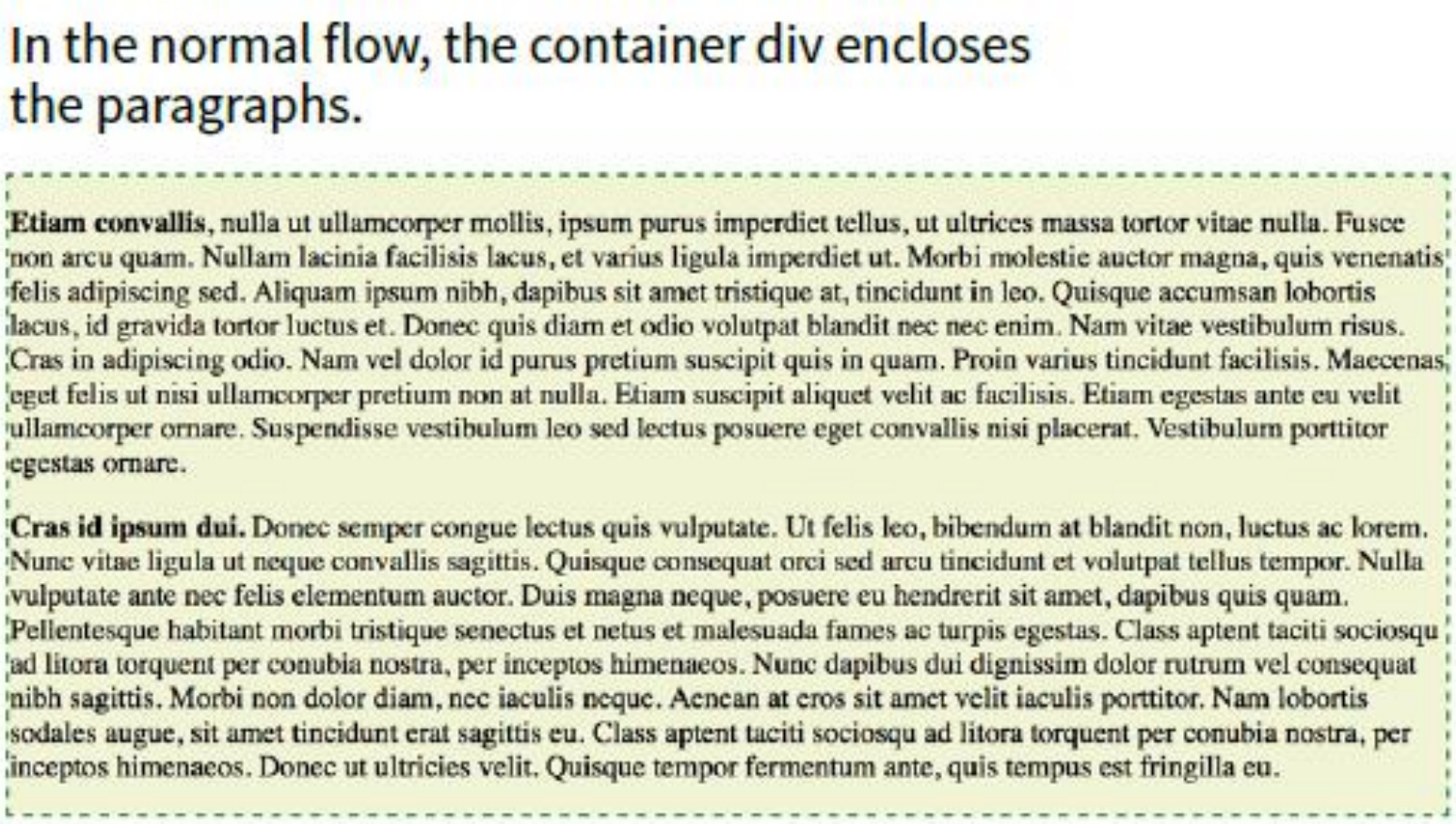
Containing Floats

```
<div id="container">
  <p>…</p>
  <p>…</p>
</div>

#container {
  background: #f2f5d5;
  border: 2px dashed green;
}

p {
  float: left;
  width: 44%;
  padding: 2%;
}
```

Containing Floats



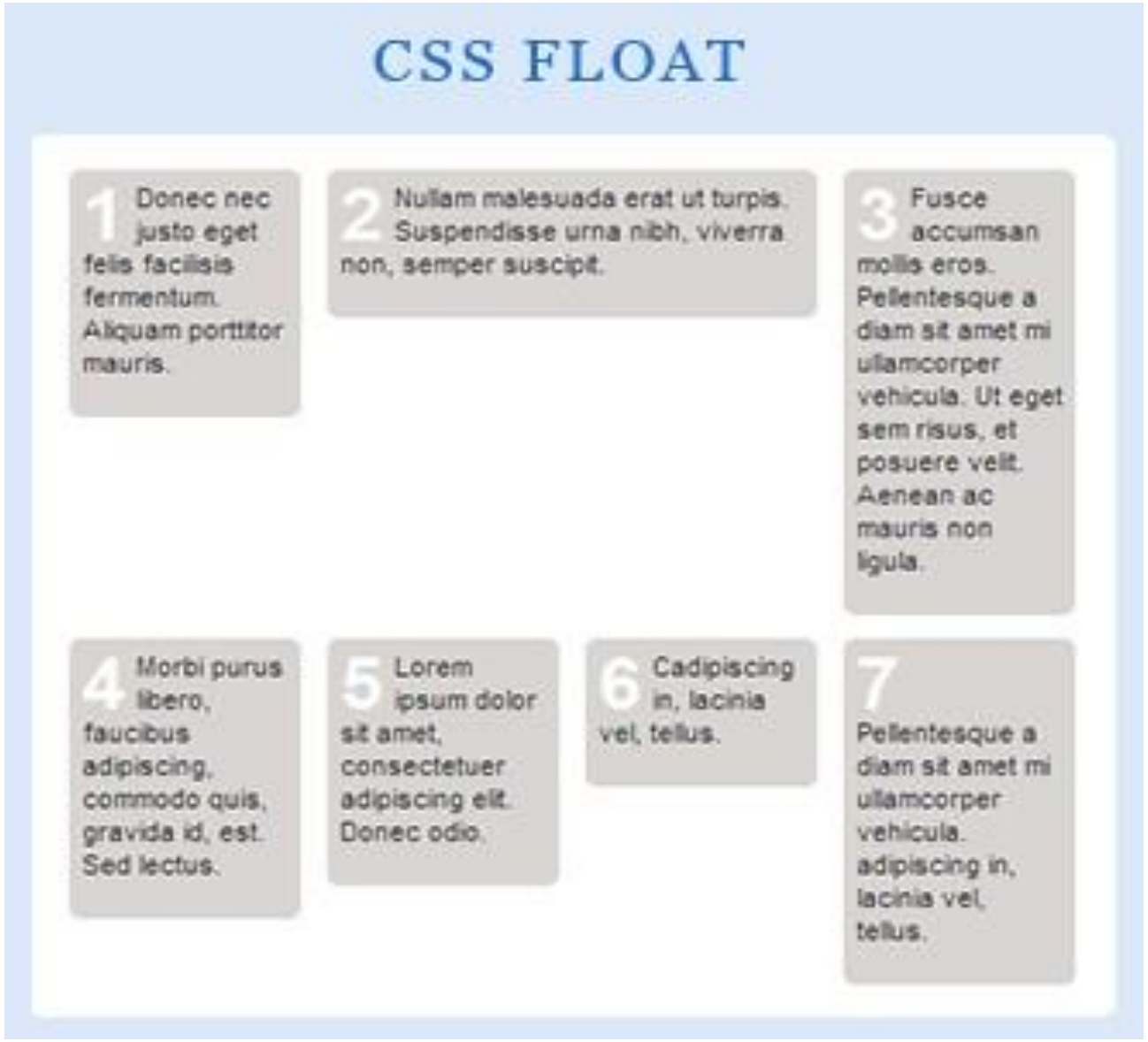
Containing Floats

When both paragraphs are floated, the container does not stretch around them.

Etiam convallis, nulla ut ullamcorper mollis, ipsum purus imperdiet tellus, ut ultrices massa tortor vitae nulla. Fusce non arcu quam. Nullam lacinia facilisis lacus, et varius ligula imperdiet ut. Morbi molestie auctor magna, quis venenatis felis adipiscing sed. Aliquam ipsum nibh, dapibus sit amet tristique at, tincidunt in leo. Quisque accumsan lobortis lacus, id gravida tortor luctus et. Donec quis diam et odio volutpat blandit nec nec enim. Nam vitae vestibulum risus. Cras in adipiscing odio. Nam vel dolor id purus pretium suscipit quis in quam. Proin varius tincidunt facilisis. Maecenas eget felis ut nisi ullamcorper pretium non at nulla. Etiam suscipit aliquet velit ac facilisis. Etiam egestas ante eu velit ullamcorper ornare. Suspendisse vestibulum leo sed lectus posuere eget convallis nisi placerat. Vestibulum porttitor egestas ornare.

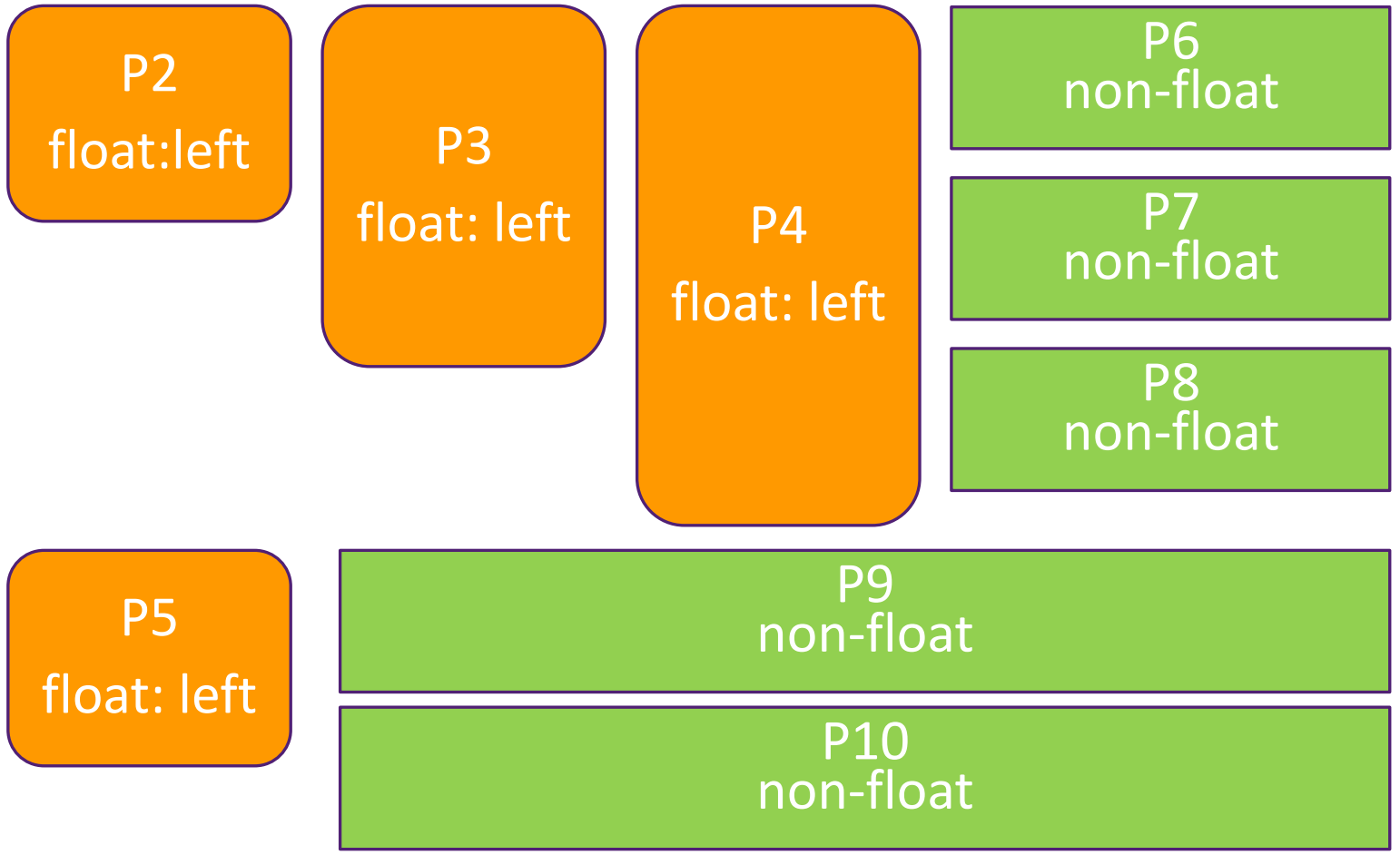
Cras id ipsum dui. Donec semper congue lectus quis vulputate. Ut felis leo, bibendum at blandit non, luctus ac lorem. Nunc vitae ligula ut neque convallis sagittis. Quisque consequat orci sed arcu tincidunt et volutpat tellus tempor. Nulla vulputate ante nec felis elementum auctor. Duis magna neque, posuere eu hendrerit sit amet, dapibus quis quam. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos. Nunc dapibus dui dignissim dolor rutrum vel consequat nibh sagittis. Morbi non dolor diam, nec iaculis neque. Aenean at eros sit amet velit iaculis porttitor. Nam lobortis sodales augue, sit amet tincidunt erat sagittis eu. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos. Donec ut ultricies velit. Quisque tempor fermentum ante, quis tempus est fringilla eu.

FIGURE 15-10. The container box disappears entirely when all its contents are floated.



CSS Float following the line to float to the left.

Non-floats fill in the empty space if following the flow.



Containing Floats

FIGURE 15-11. Our hanging floats are now contained.

Containing Floats

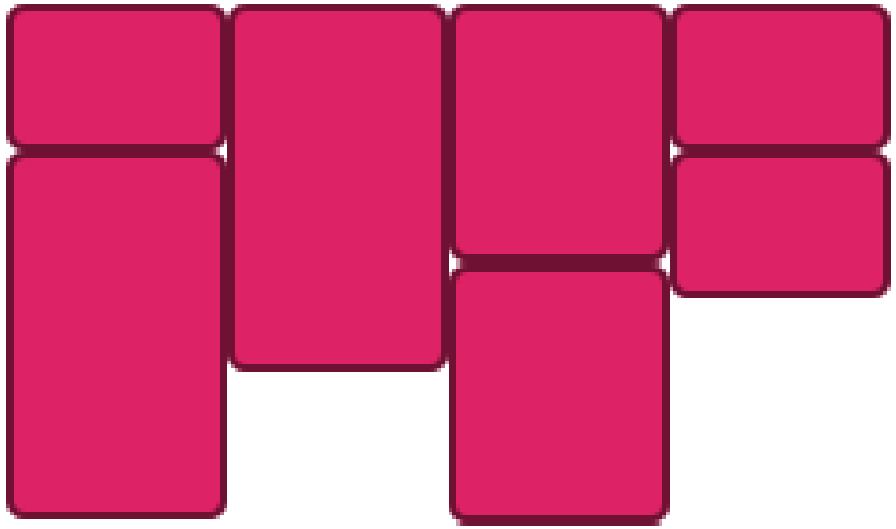
- Here it is applied to the **#container div** in FIGURE 15-10:

```
#container:after {
    content: " ";
    display: block;
    clear: both;
    background-color: #f2f5d5; /*light
    green*/
    border: 2px dashed green;
    padding: 1em;
}
```

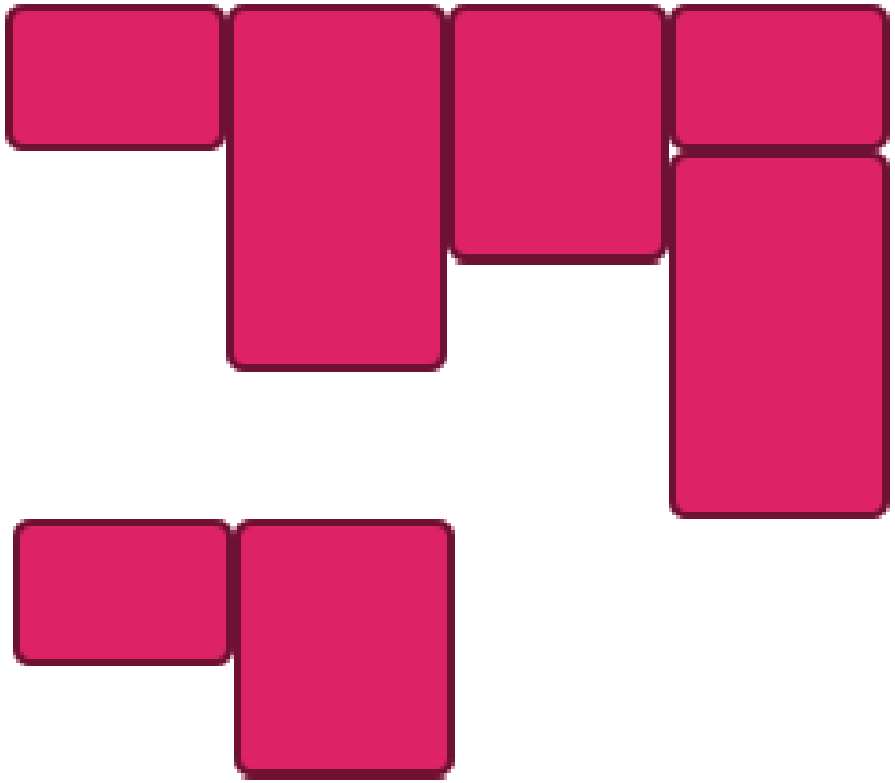
- Another option is to float the containing element as well and give it a width of 100%:

```
#container {
    float: left;
    width: 100%;
    …
}
```

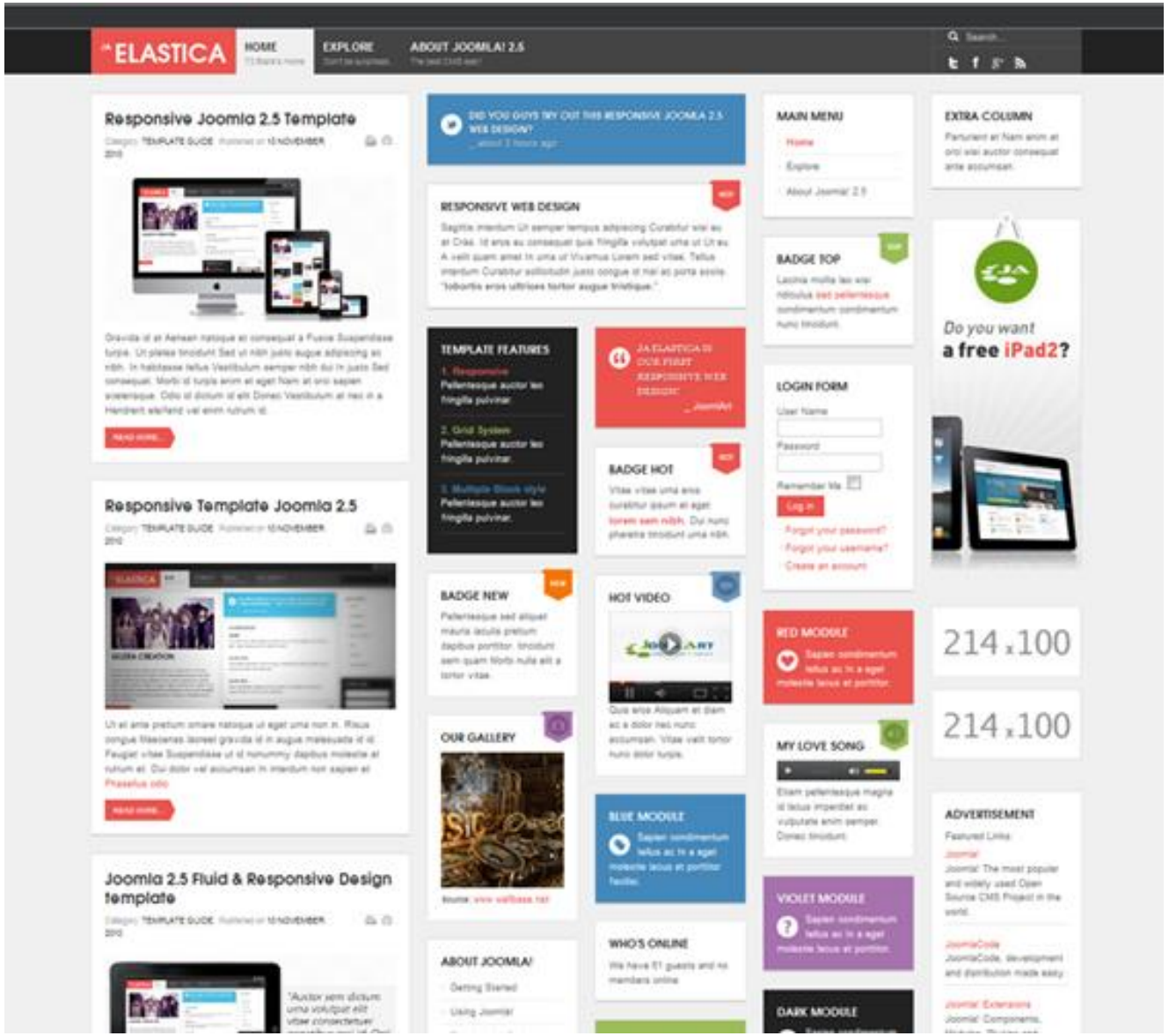
What you want



What you most likely get



What can still be improved for floating multiple elements



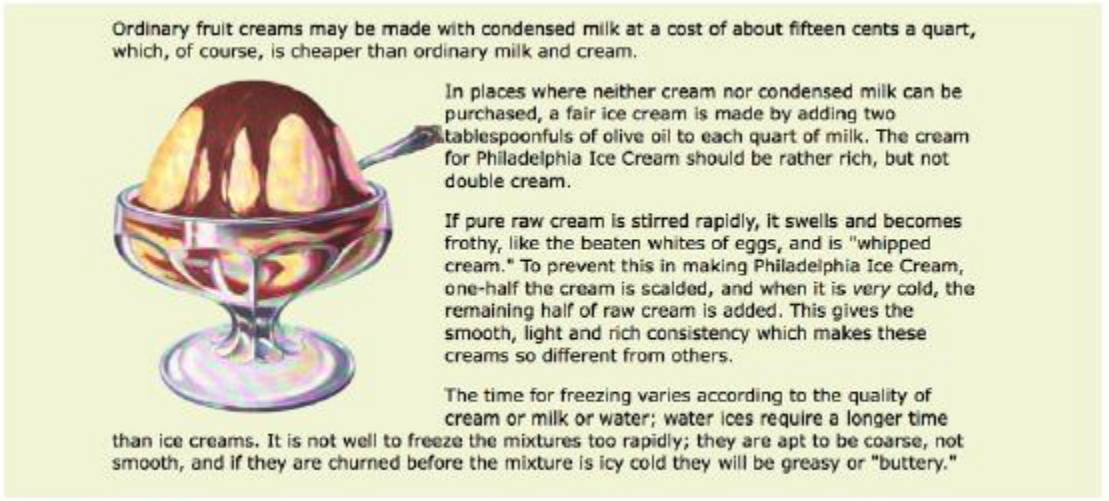
A Masonry Page

CSS Shapes

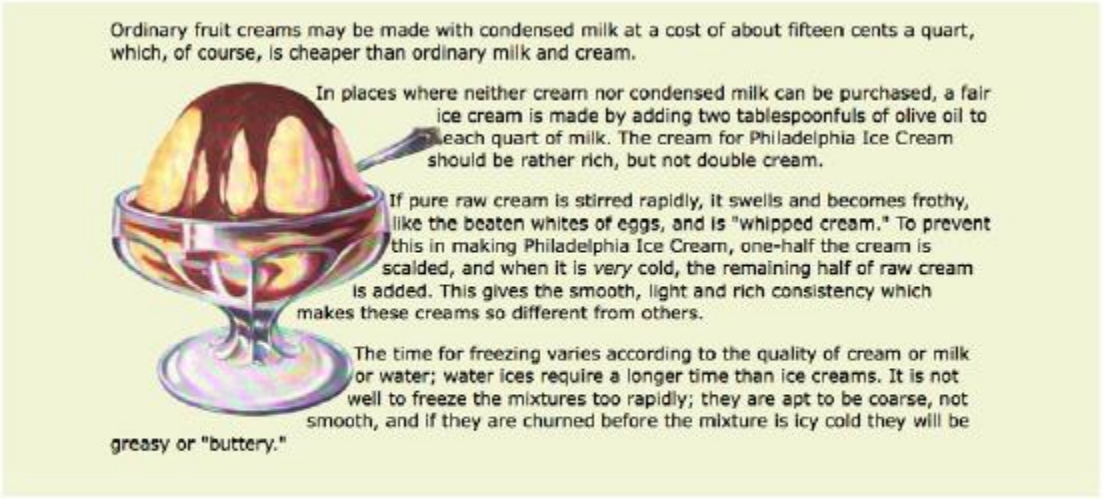
For Fancy Text Wraps

SECTION 8

Fancy Text Wraps with CSS Shapes



Default text wrap



Text wrap with `shape-outside` using the transparent areas of the image as a guide

FIGURE 15-12. Example of text wrapping around an image with **shape-outside**.

Fancy Text Wraps with CSS Shapes

- FIGURE 15-12 shows the default text wrap around a floated image (left) and the same wrap with **shape-outside** applied (right). This is the kind of thing you'd expect to see in a print magazine, but now we can do it on the web!
- It is worth noting that you can change the text wrap shape around any floated element (see Note), but I will focus on images in this discussion, as text elements are generally boxes that fit nicely in the default rectangular wrap.
- There are two approaches to making text wrap around a shape. One way is to provide the path coordinates of the wrap shape with **circle()**, **ellipse()**, or **polygon()**. Another way is to use **url()** to specify an image that has transparent areas (such as a GIF or a PNG). With the image method, text flows into the transparent areas of the image and stops at the opaque areas. This is the shape method shown in FIGURE 15-12 and the method I'll introduce first.

Fancy Text Wraps with CSS Shapes

shape-outside

Values:	none | circle() | ellipse() | polygon() | url() | [margin-box | padding-box | content-box]
Default:	none
Applies to:	floats
Inherits:	no

shape-margin

Values:	*length* | *percentage*
Default:	0
Applies to:	floats
Inherits:	no

Using a Transparent Image

THE MARKUP

```
<p><img src="sundae.png" class="wrap"
alt=""> In places…</p>
```

THE STYLES

```
img.wrap {
    float: left;
    width: 300px;
    height: 300px;
    -webkit-shape-outside:
    url(sundae.png); /* prefix required
    in 2018 */
    shape-outside: url(sundae.png);

    -webkit-shape-margin: 1em;
    shape-margin: 1em;
}
```

Using a Transparent Image

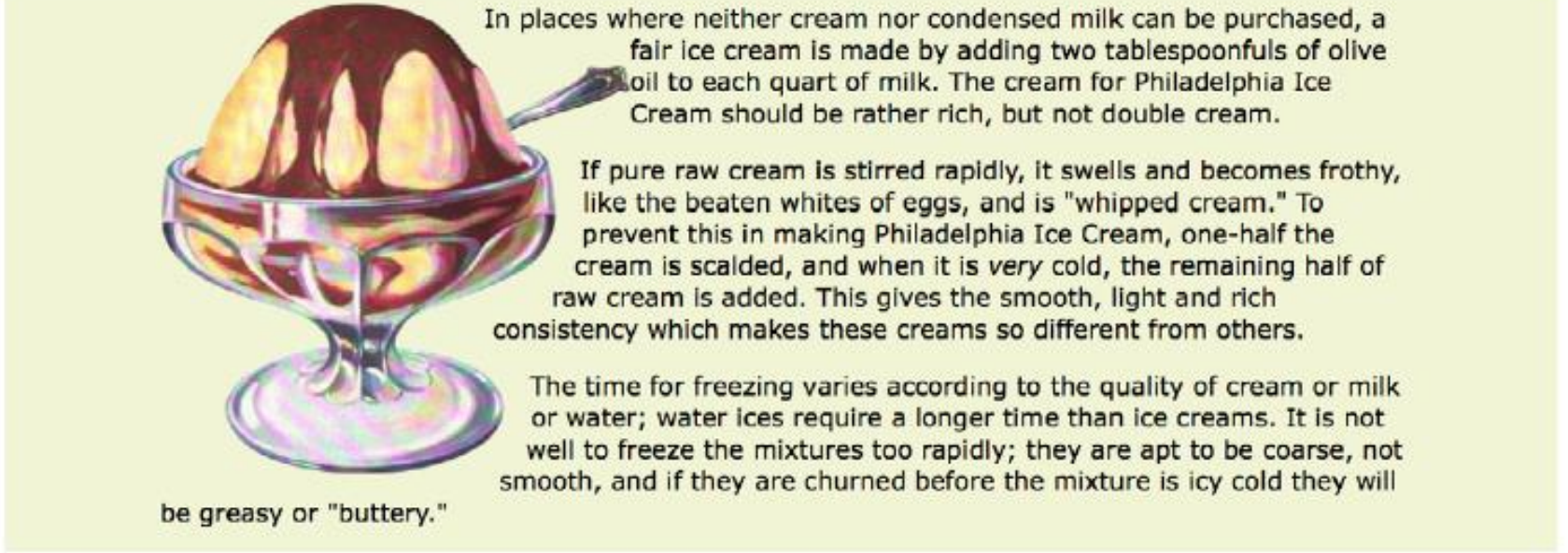


FIGURE 15-13. Adding a margin between the shape and the wrapped text.

Using a Path

- The other method for creating a text wrap shape is to define it using one of the path keywords: **circle()**, **ellipse()**, and **polygon()**.
- The **circle()** notation creates a circle shape for the text to wrap around. The value provided within the parentheses represents the length of the radius of the circle:

`circle(`*`radius`*`)`

Using a Path

- In this example, the radius is 150px, half of the image width of 300 pixels. By default, the circle is centered vertically and horizontally on the float:

```
img.round {
   float: left;
   -webkit-shape-outside:
   circle(150px);
   shape-outside: circle(150px);
}
```

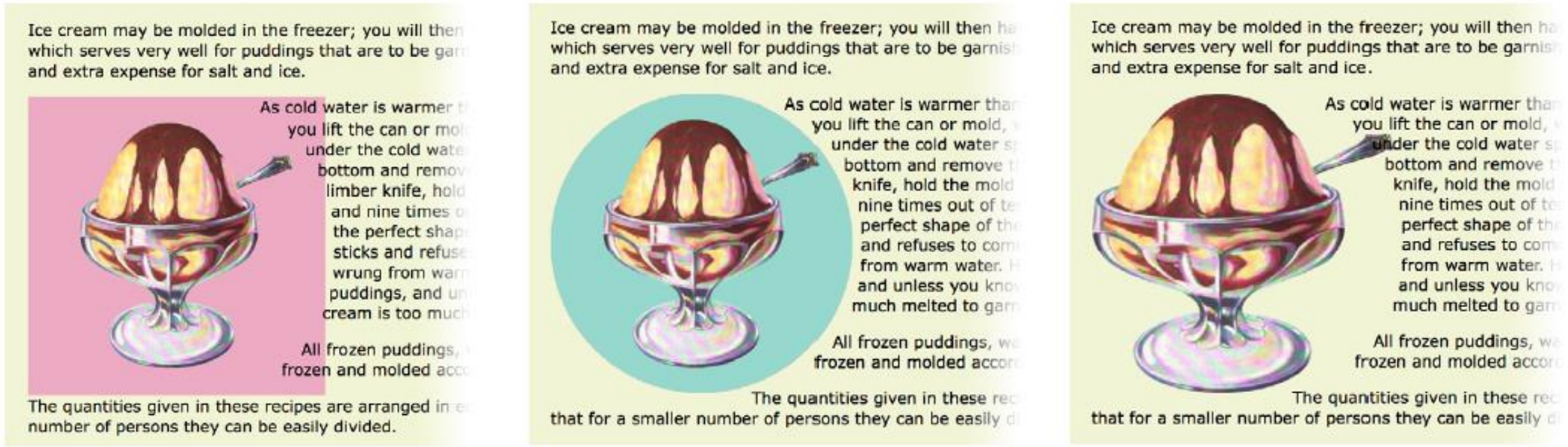


FIGURE 15-14. The same `circle()` shape applied to different images in the source.

Using a Path

Using a Path

- In the example in FIGURE 15-15, I've increased the diameter of the circle path from 150px to 200px. Notice that the text lines up along the right edge of the image, even though the circle is set 50 pixels beyond the edge. The path does not push text away from the float. If you need to keep wrapped text away from the outside edge of the floated image or element, apply a margin to the element itself (it will be the standard rectangular shape, of course).

Using a Path

```
img.round {
  float: left;
  -webkit-shape-outside: circle(200px);
  shape-outside: circle(200px);
}
```

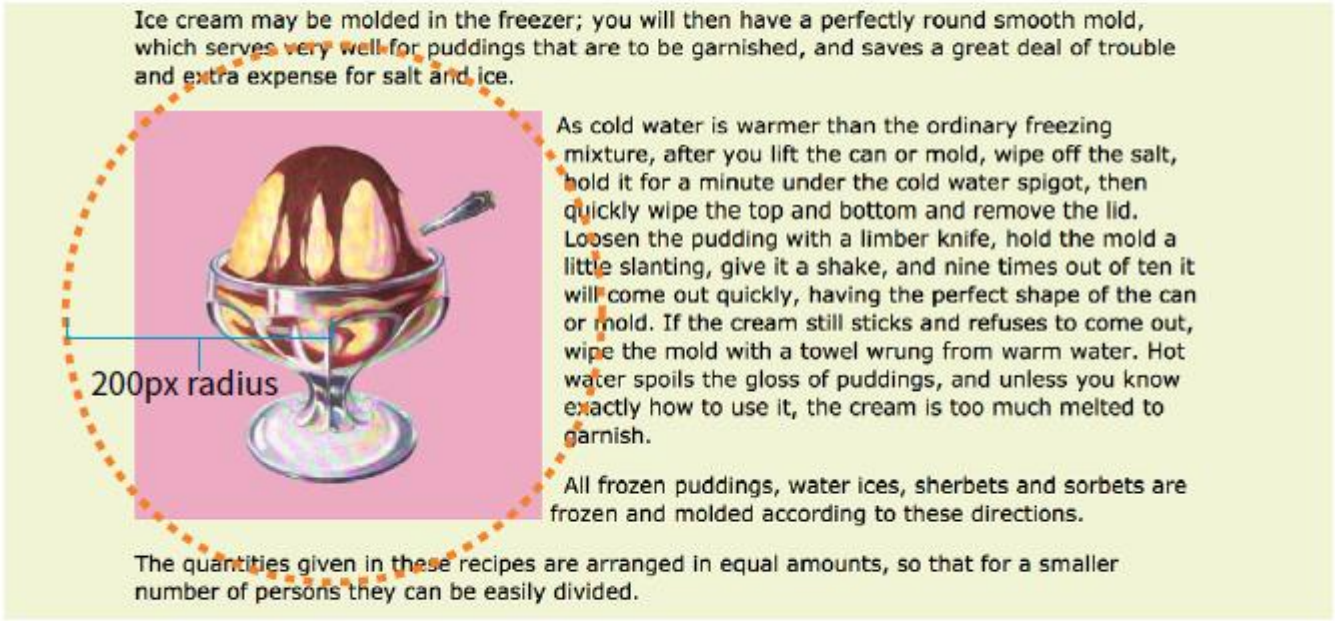


FIGURE 15-15. CSS shapes allow text to wrap into the floated element but do not hold space beyond it.

Using a Path

ellipse()

- Elliptical shapes are created with the **ellipse()** notation, which provides the horizontal and vertical radius lengths followed by the word **at** and then the x,y coordinates for the center of the shape. Here is the syntax:

```
ellipse(rx ry at x y);
```

```
img.round {
  float: left;
  -webkit-shape-outside: ellipse(200px 100px at 50% 50%);
  shape-outside: ellipse(200px 100px at 50% 50%);
}
```

The edges of the image (blue) and ellipse path (dotted orange) revealed

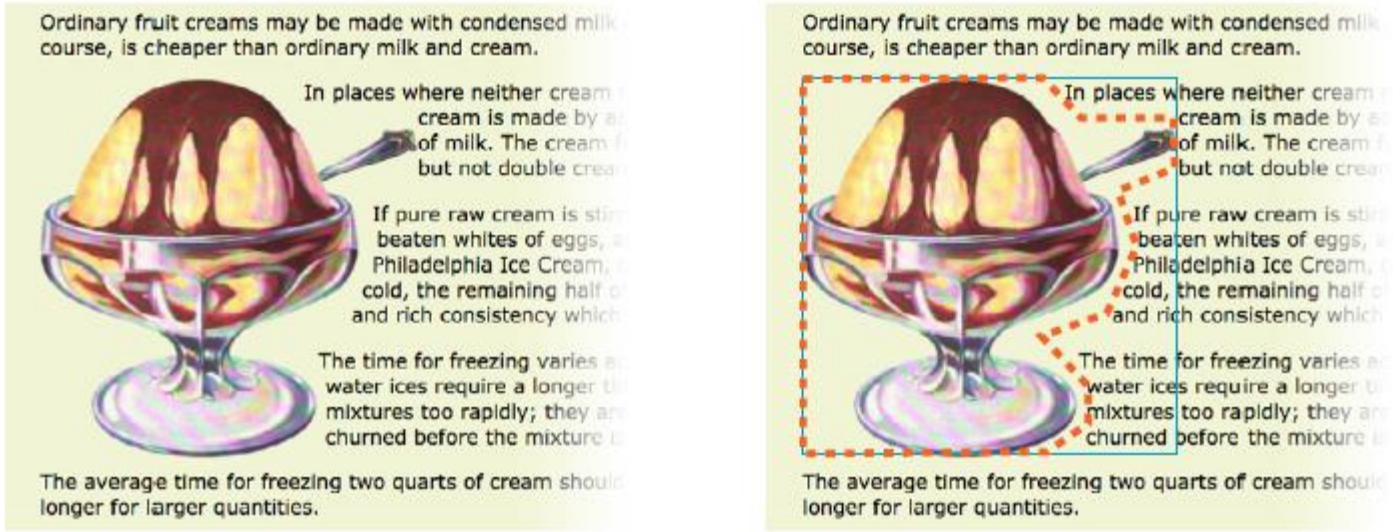
FIGURE 15-16. An elliptical text wrap created with `ellipse()`.

Using a Path

ellipse()

Finally, we come to `polygon()`, which lets you create a custom path using a series of comma-separated x,y coordinates along the path. This style rule creates the wrap effect shown in FIGURE 15-17:

```
img.wrap {
  float: left;
  width: 300px;
  height: 300px;
  shape-outside: polygon(0px 0px, 186px 0px, 225px 34px, 300px 34px,
300px 66px, 255px 88px, 267px 127px, 246px 178px, 192px 211px, 226px
236px, 226px 273px, 209px 300px, 0px 300px);
}
```



The edges of the image (blue) and polygon path (dotted orange) revealed

FIGURE 15-17. A custom path created with `polygon()`.

Using a Path

polygon()

Positioning

SECTION 9

Positioning

position

Values:	static | relative | absolute | fixed
Default:	static
Applies to:	all elements
Inherits:	no

Positioning

static: Positioned as they occur in the normal flow. (default)

relative: relative positioning moves the box **relative to its original position** in the flow.

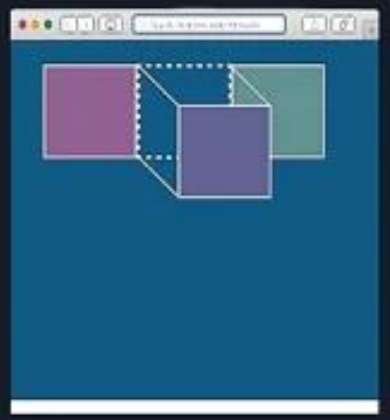
absolute: Absolutely positioned elements are removed from the document flow entirely and positioned **with respect to the browser window or a containing element**.

fixed: The element **stays** in one position **in the window** even when the document scrolls.

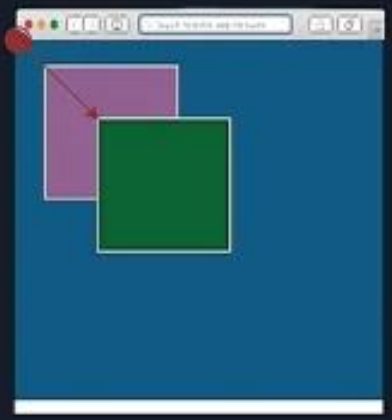
sticky (new): Sticky positioning is a combination of relative and fixed in that it behaves as though it is relatively positioned, until it is scrolled into a specified position relative to the viewport, at which point it remains fixed.

CSS Position Property

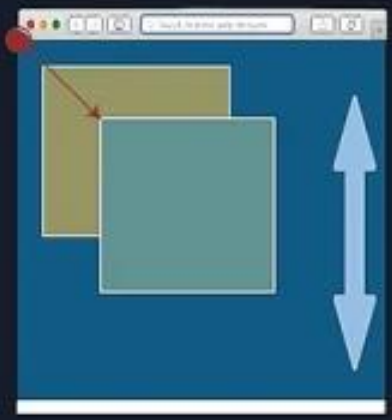
Static



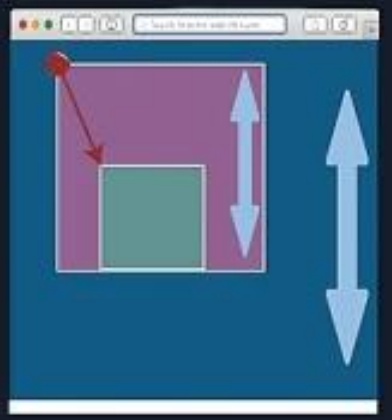
Relative



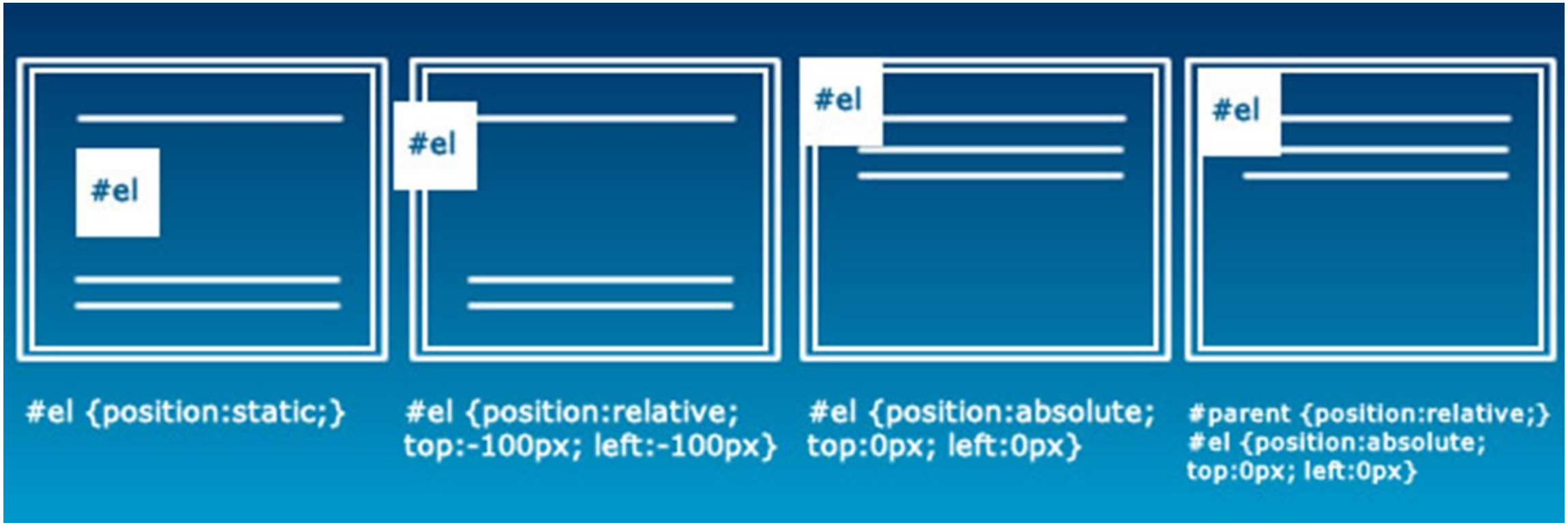
Absolute



Fixed



Sticky


#el
#el
#el
#el
#el {position:static;}
#el {position:relative;
top:-100px; left:-100px}
#el {position:absolute;
top:0px; left:0px}
#parent {position:relative;}
#el {position:absolute;
top:0px; left:0px}

Specifying Position

Once you've established the positioning method, the actual position is specified with some combination of up to four offset properties.

top, **right**, **bottom**, **left**

Values:	*length* | *percentage* | auto
Default:	auto
Applies to:	positioned elements (where position value is relative, absolute, or fixed)
Inherits:	no

```
em {
  position: relative;
  top: 2em; /* moves element down */
  left: 3em; /* moves element right */
  background-color: fuchsia;
}
```

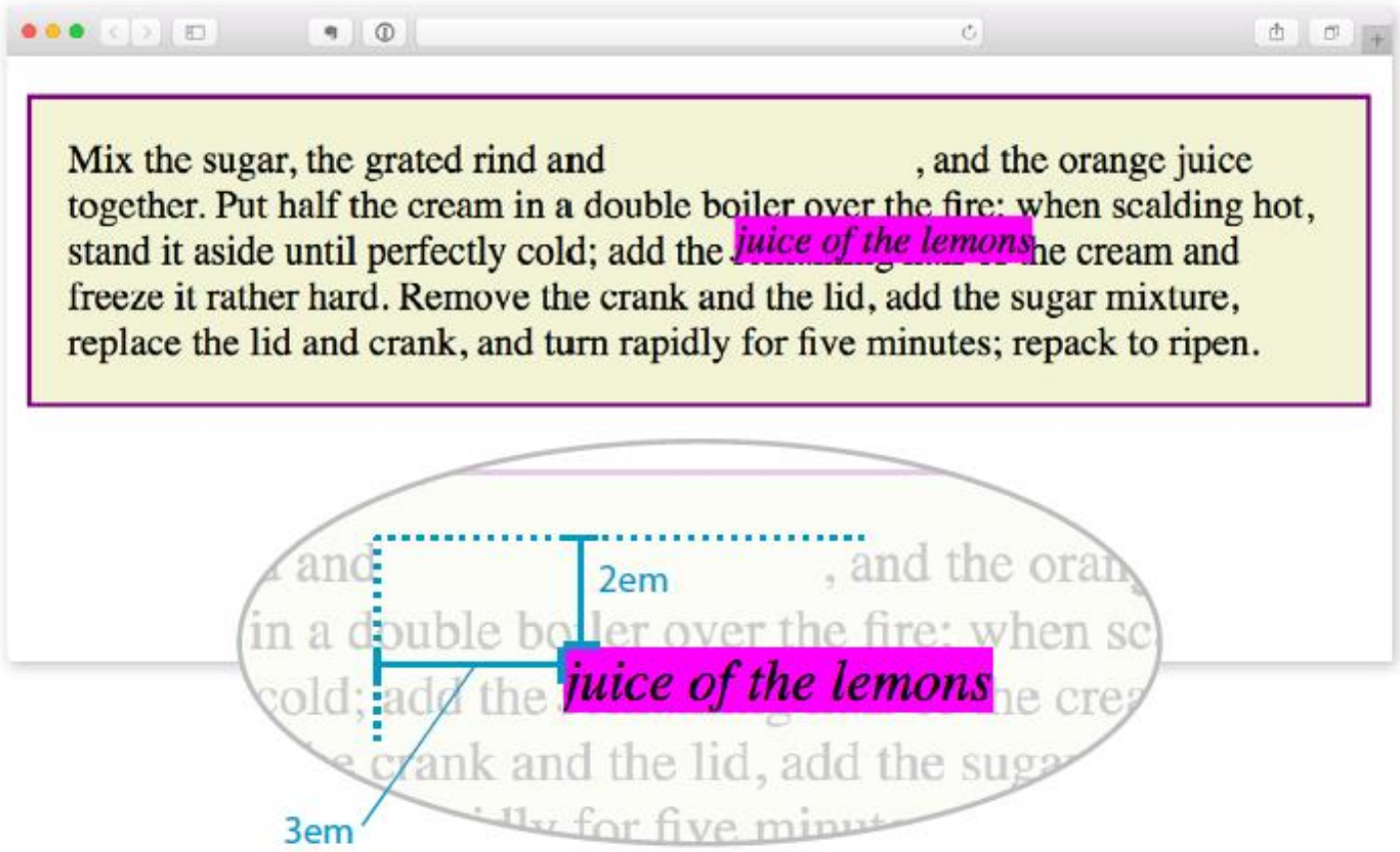


FIGURE 15-19. When an element is positioned with the relative method, the space it would have occupied is preserved.

Relative Positioning

Relative positioning

```
div.b{
   position:relative;
   top:20px;
   left:20px;
   }
```

```
div.b {
        position:relative;
        top:50px;
        left:20px;
        }
```

- new position of the element
- old position of the element

```
em {
  position: absolute;
  top: 2em;
  left: 3em;
  background-color: fuchsia;
}
```

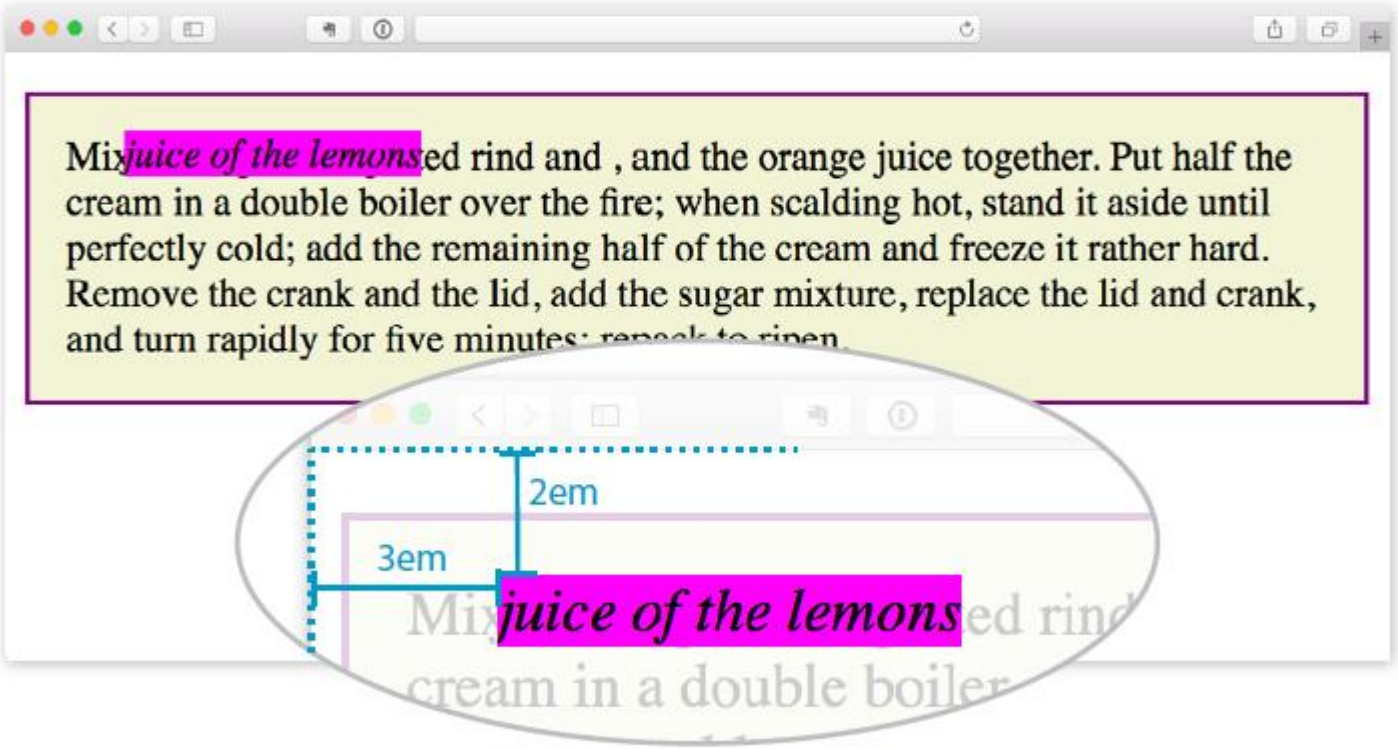
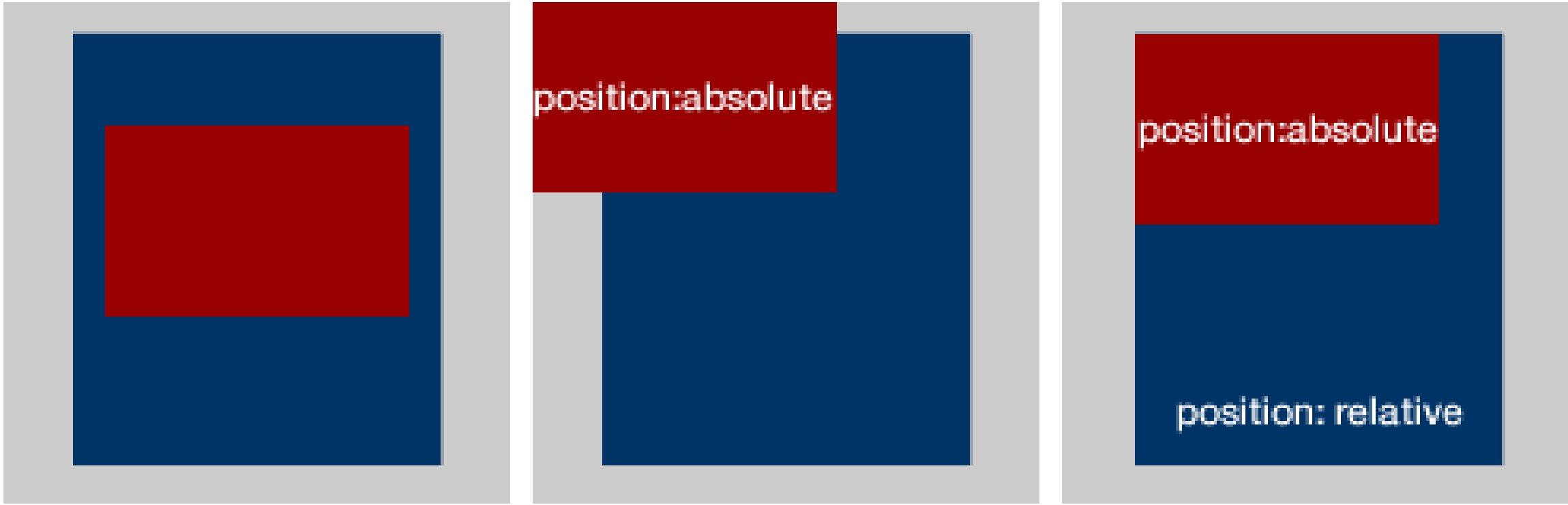


FIGURE 15-20. When an element is absolutely positioned, it is removed from the flow and the space is closed up.

Absolute Positioning



Another look at absolute to parent and absolute to window.

Absolute to Window (no relative parent)

```
div.c{
   position: absolute;
   top:20px;
   left:20px;
   }
```

```
<div class="b">
    <div class="c"></div>
</div>
```

```
div.b{position: relative; }
div.c{
    position: absolute;
    top:20px;
    left:20px;
    }
```

Absolute Positioning With Relative Parent

```
p {
  position: relative;
  padding: 15px;
  background-color: #F2F5D5;
  border: 2px solid purple;
}
```

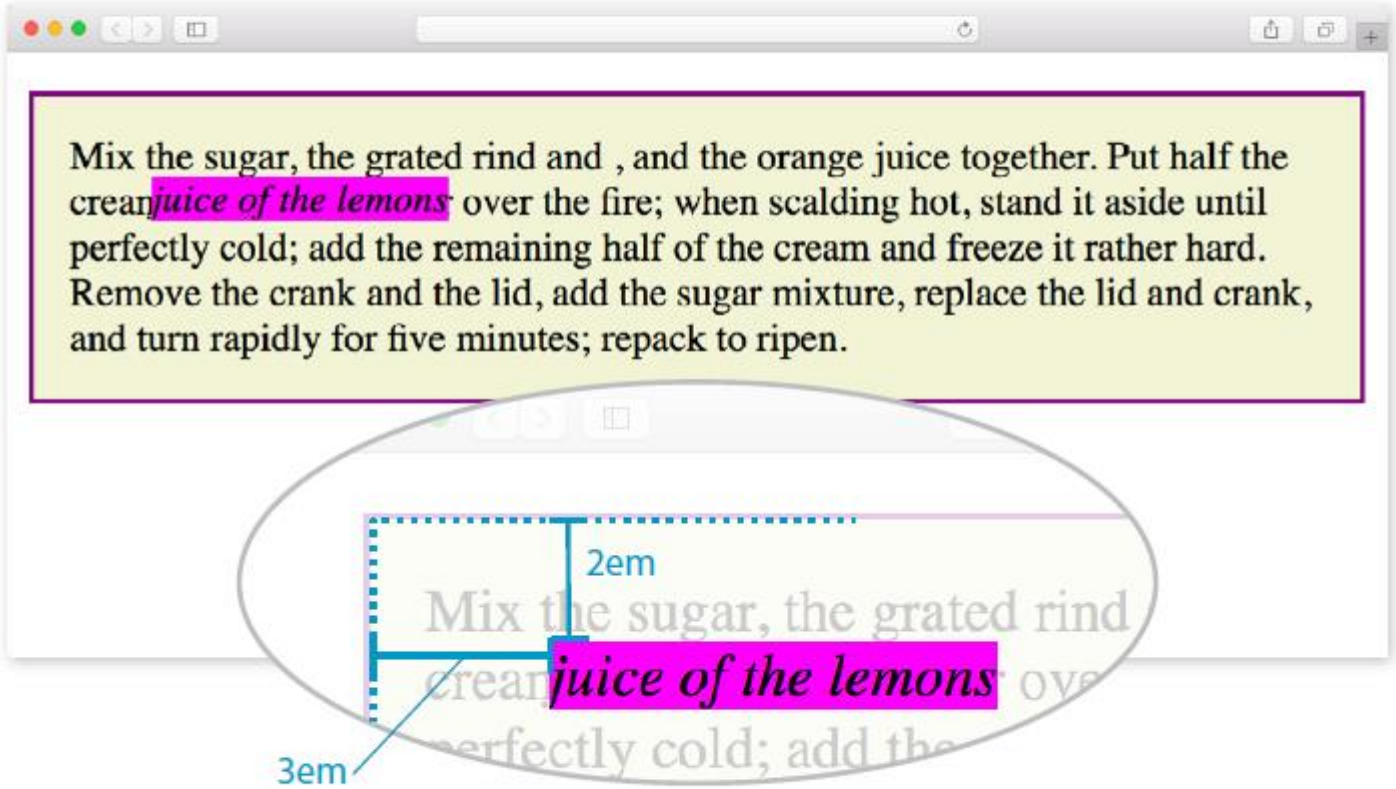


FIGURE 15-21. The relatively positioned p element acts as a containing block for the em element.

Absolute Positioning With Relative Parent

```
em {
  width: 200px;
  margin: 25px;
  position: absolute;
  top: 2em;
  left: 3em;
  background-color: fuchsia;
}
```

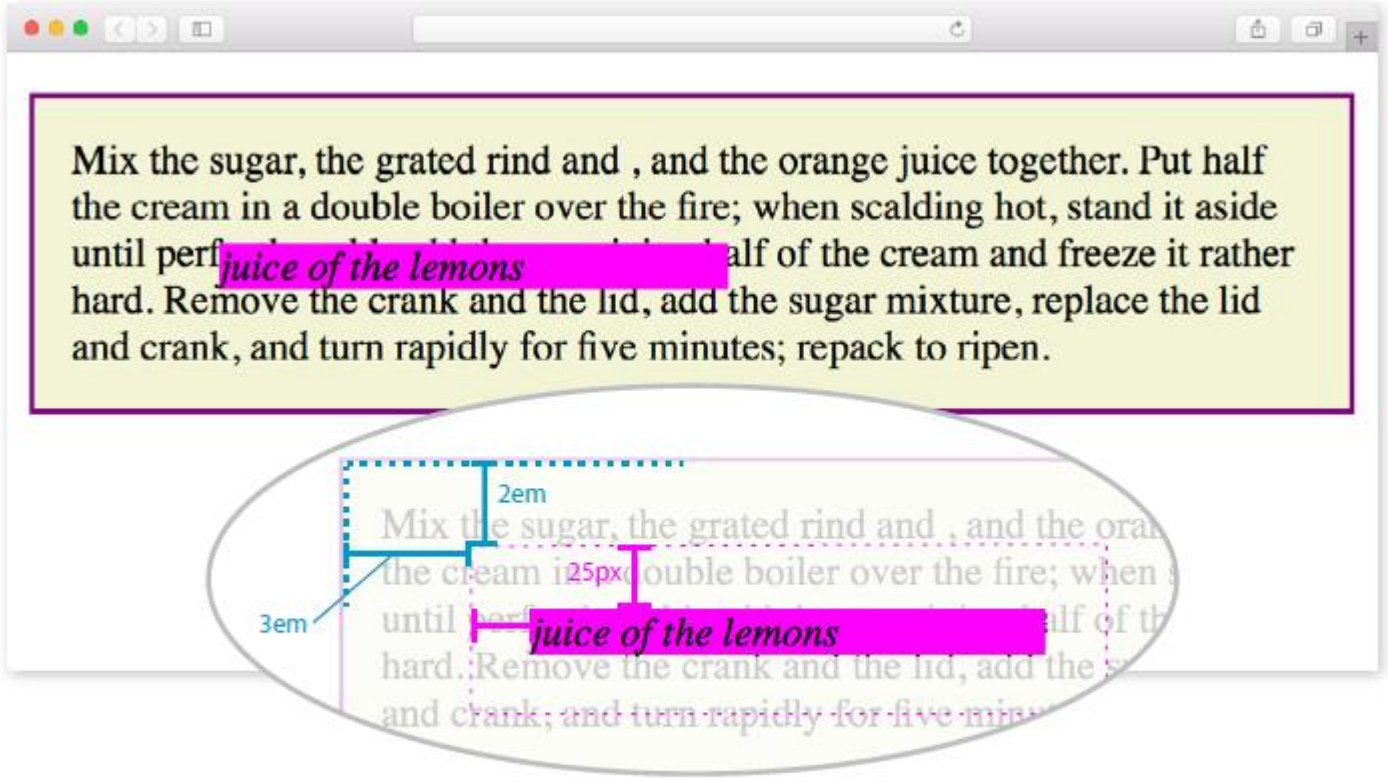


FIGURE 15-22. Adding a width and margins to the positioned element.

Absolute Positioning With Relative Parent

Specifying Position

Pixel Measurement

THE MARKUP

```
<div id="A">
<div id="B"> </div>
</div>
```

THE STYLES

```
div#A {
    position: relative; /* creates the
    containing block */
    width: 600px;
    height: 300px;
    background-color: #C6DE89; /* green */
}
div#B {
    position: absolute;
    top: 25px;
    right: 50px;
    bottom: 75px;
    left: 100px;
    background-color: steelblue;
}
```

Specifying Position

Pixel Measurement

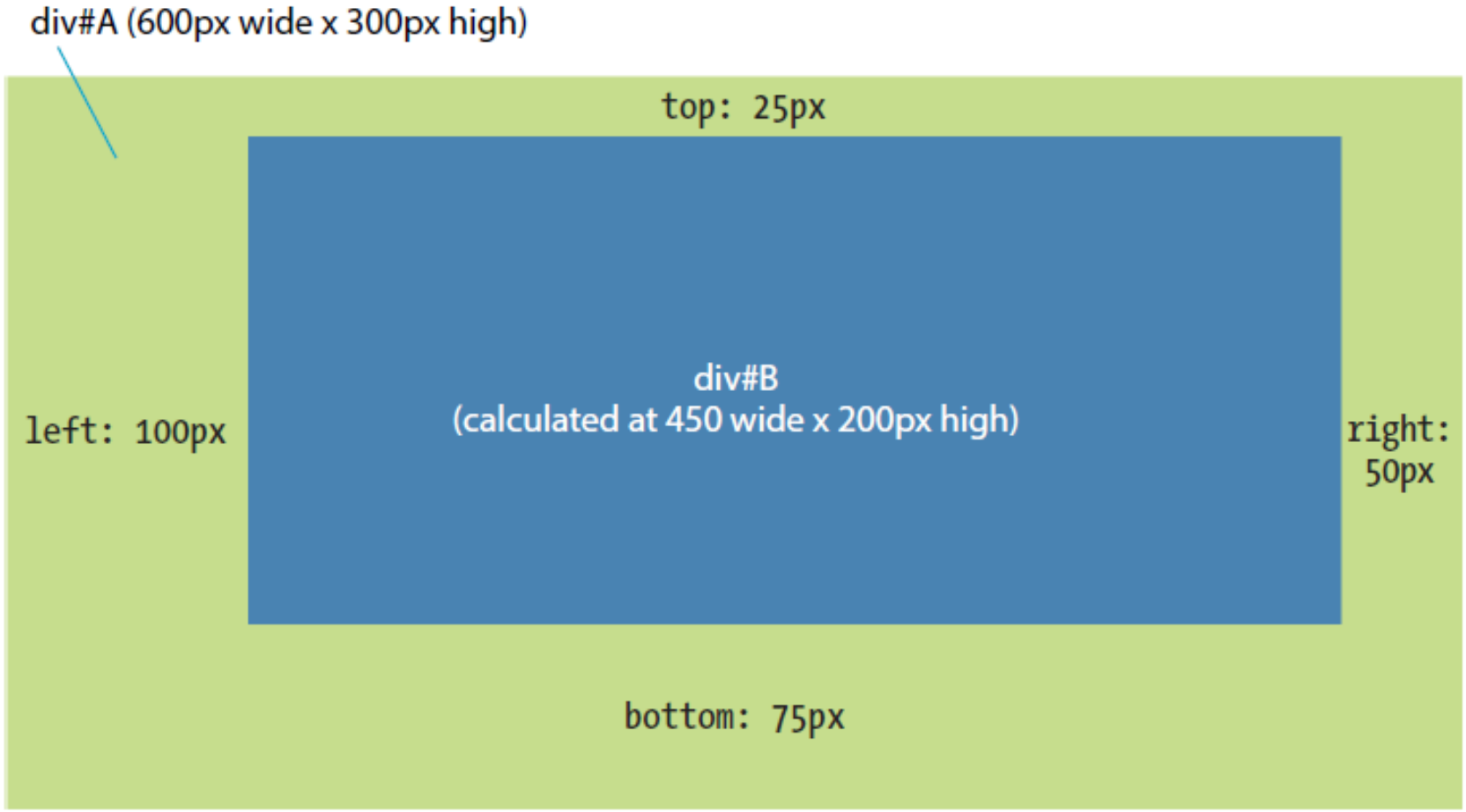


FIGURE 15-23. Setting offset values for all four sides of a positioned element.

Specifying Position

Percentage values

```
img#A {
   position: absolute;
   top: 50%;
   left: 0%;
}

img#B {
   position: absolute;
   bottom: 0%;
   right: 0%;
}
```

Specifying Position

Percentage values

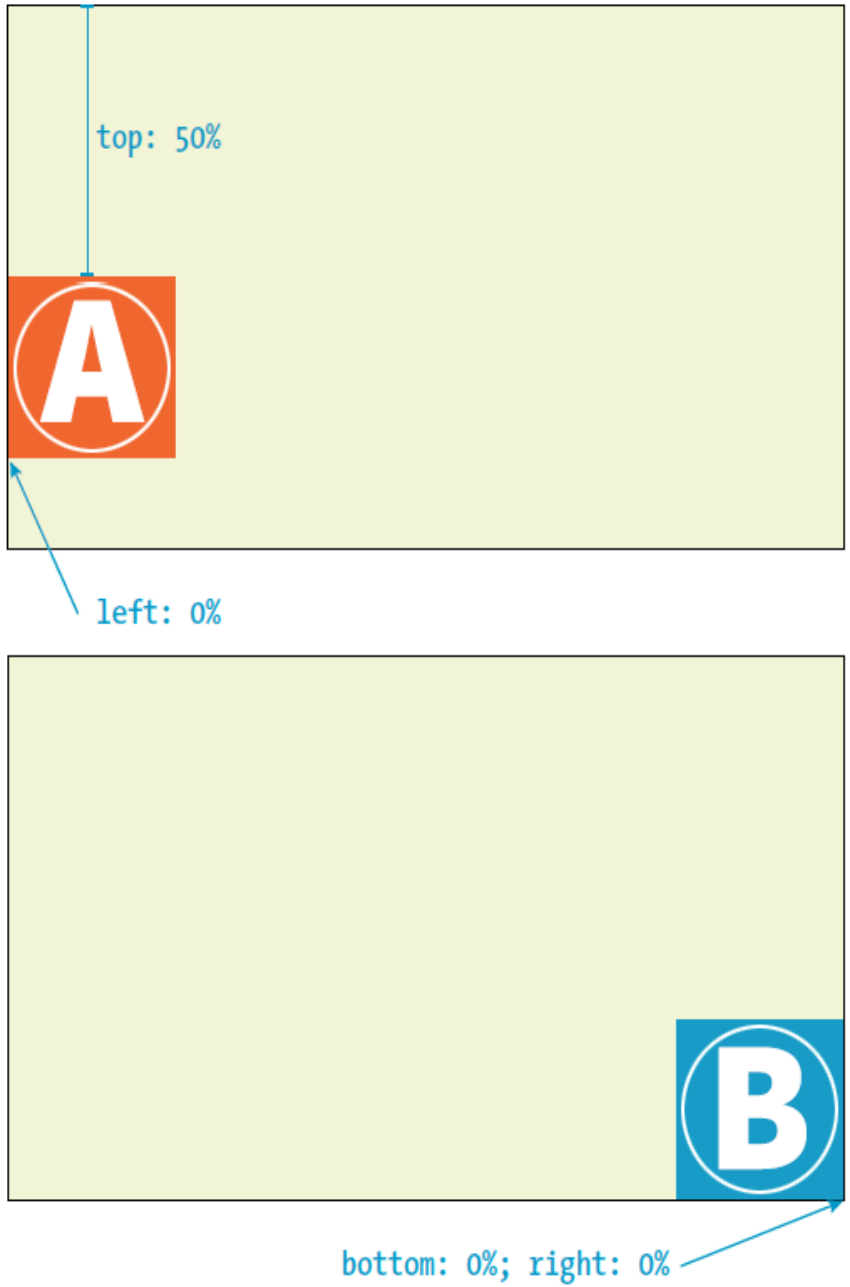


FIGURE 15-24. Using percentage values to position an element in a containing block.

Stack Order and Z-index

SECTION 10

Stacking order

z-index: number | auto | inherit

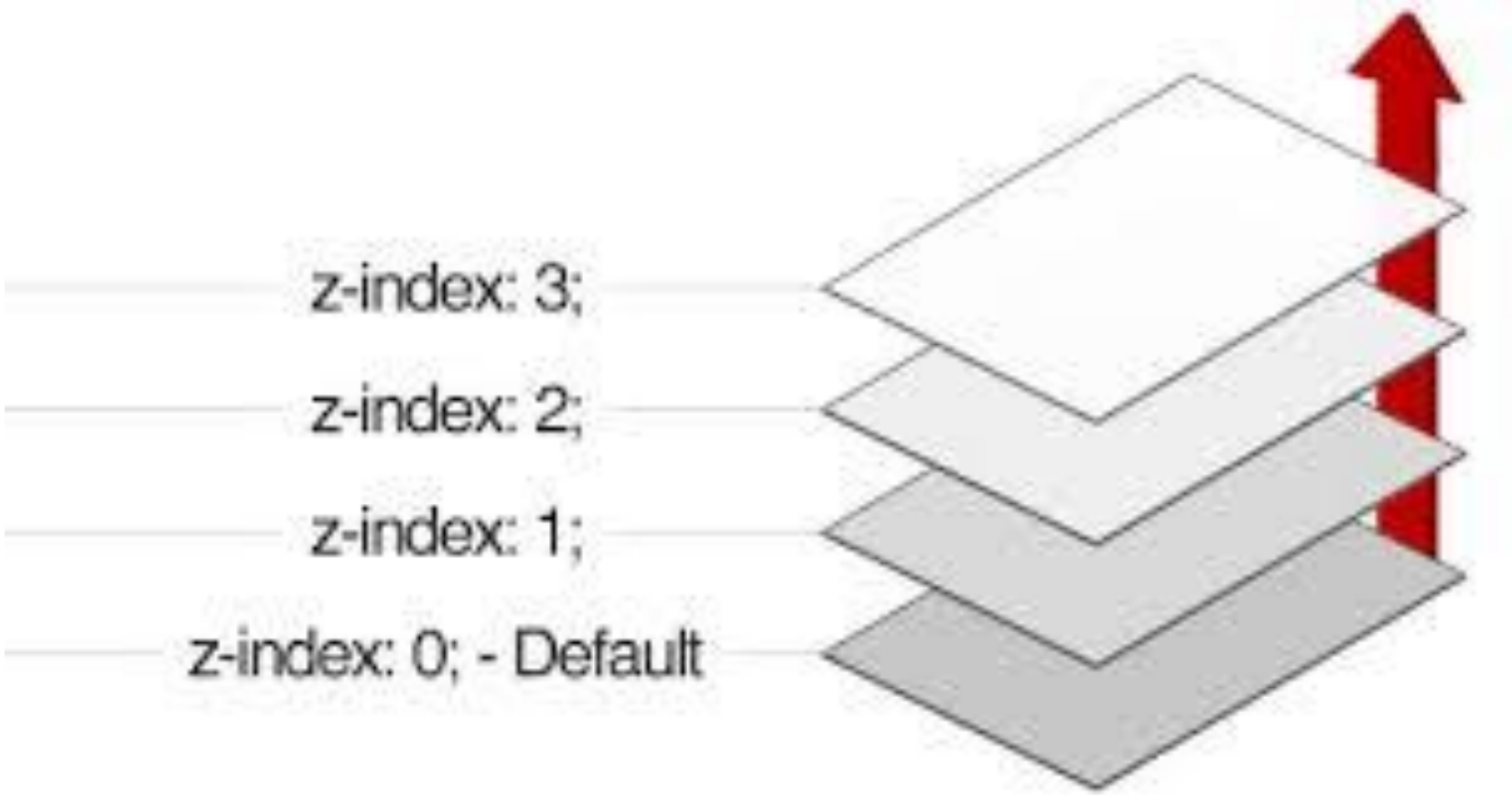
- By default, elements stack up in the order in which they appear in the document, but you can change the stacking order with the **z-index** property (see Note). Picture the z-axis as a line that runs perpendicular to the page, as though from the tip of your nose, through this age, and out the other side.

z-index

Values:	*number* | auto
Default:	auto
Applies to:	positioned elements
Inherits:	no

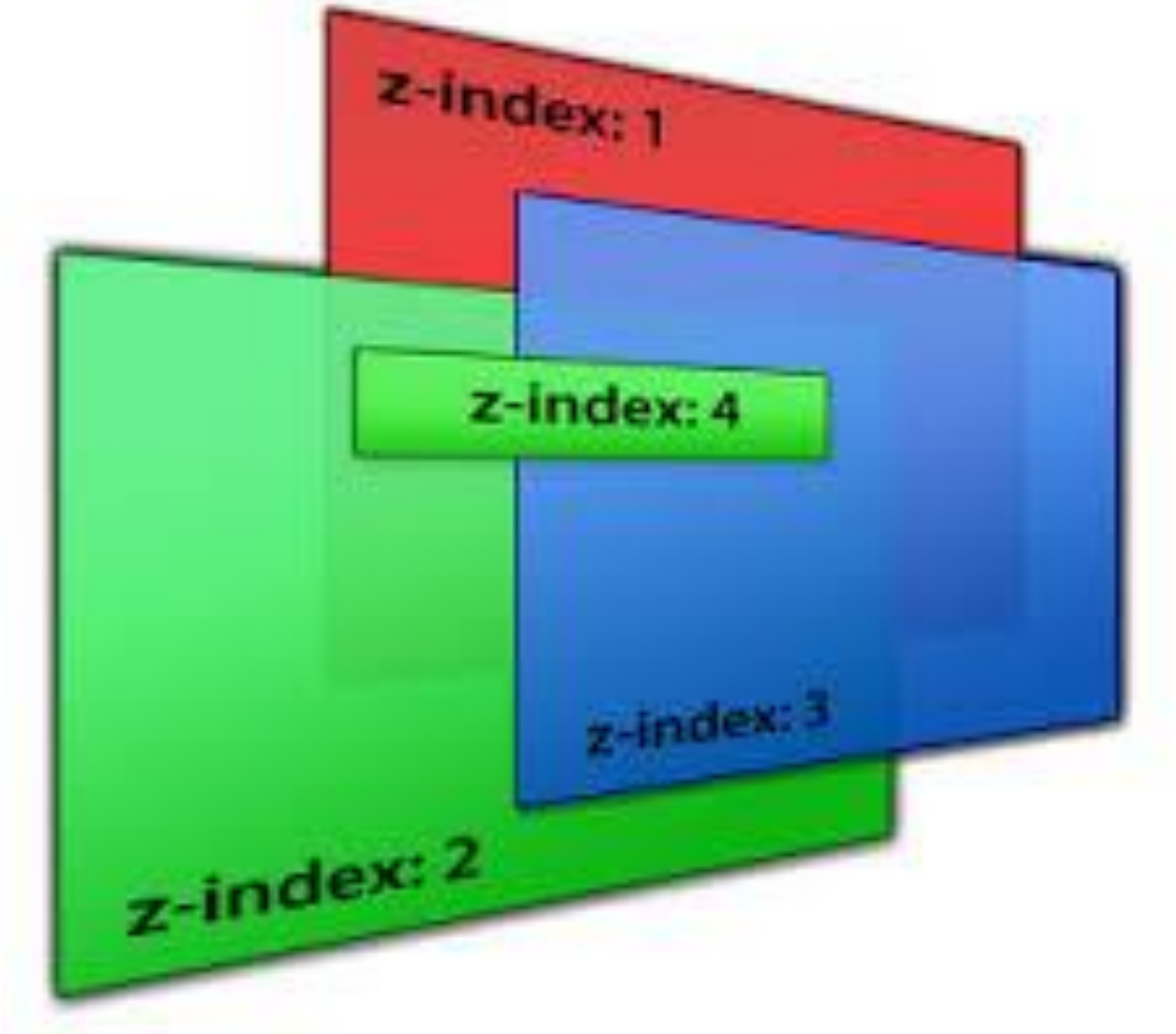
Stacking order

z-index: number |
auto | inherit



Stacking order

z-index: number |
auto | inherit



Stacking order

z-index: number |
auto | inherit

THE MARKUP

```
<p id="A"><img src="A.gif" alt="A"></p>
<p id="B"><img src="B.gif" alt="B"></p>
<p id="C"><img src="C.gif" alt="C"></p>
```

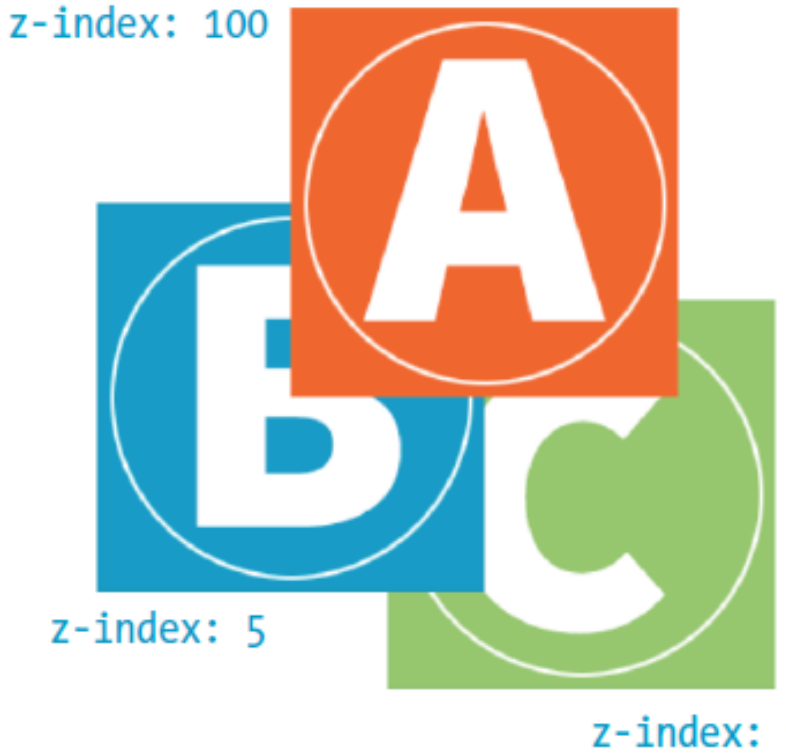
THE STYLES

```
#A {
    z-index: 100;
    position: absolute;
    top: 175px;
    left: 200px;
}
#B {
    z-index: 5;
    position: absolute;
    top: 275px;
    left: 100px;
}
#C {
    z-index: 1;
    position: absolute;
    top: 325px;
    left: 250px;
}
```

Stacking order

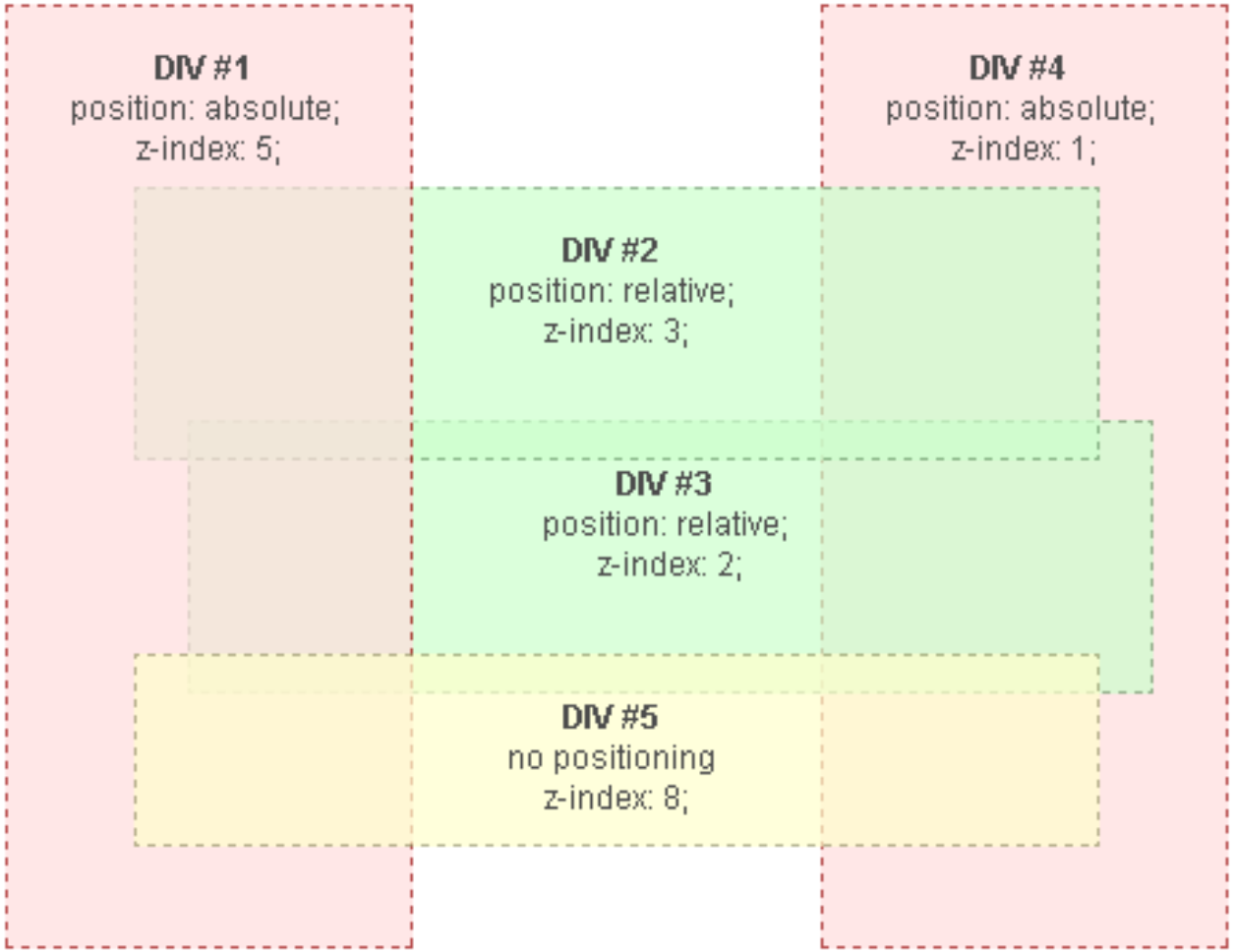
z-index: number | auto| inherit

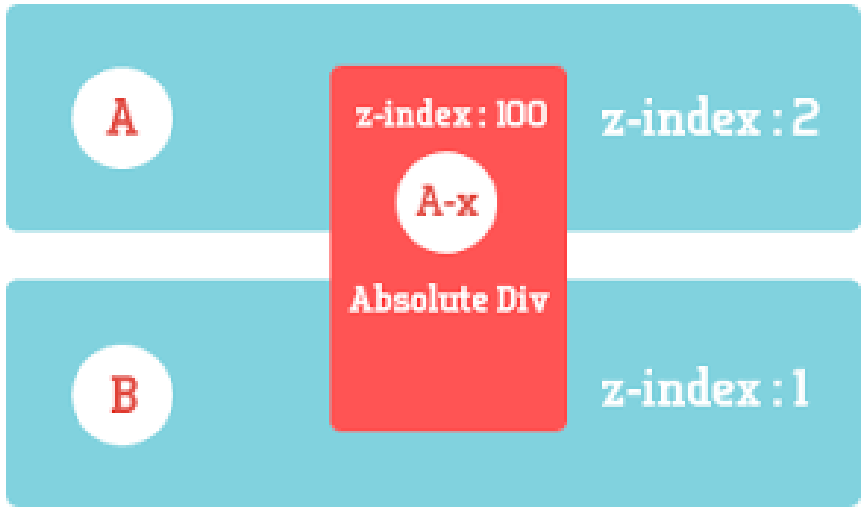
By default, elements later in the document source order stack on top of preceding elements.



You can change the stacking order with the z-index property. Higher values stack on top of lower values.

FIGURE 15-26. Changing the stacking order with the **z-index** property.





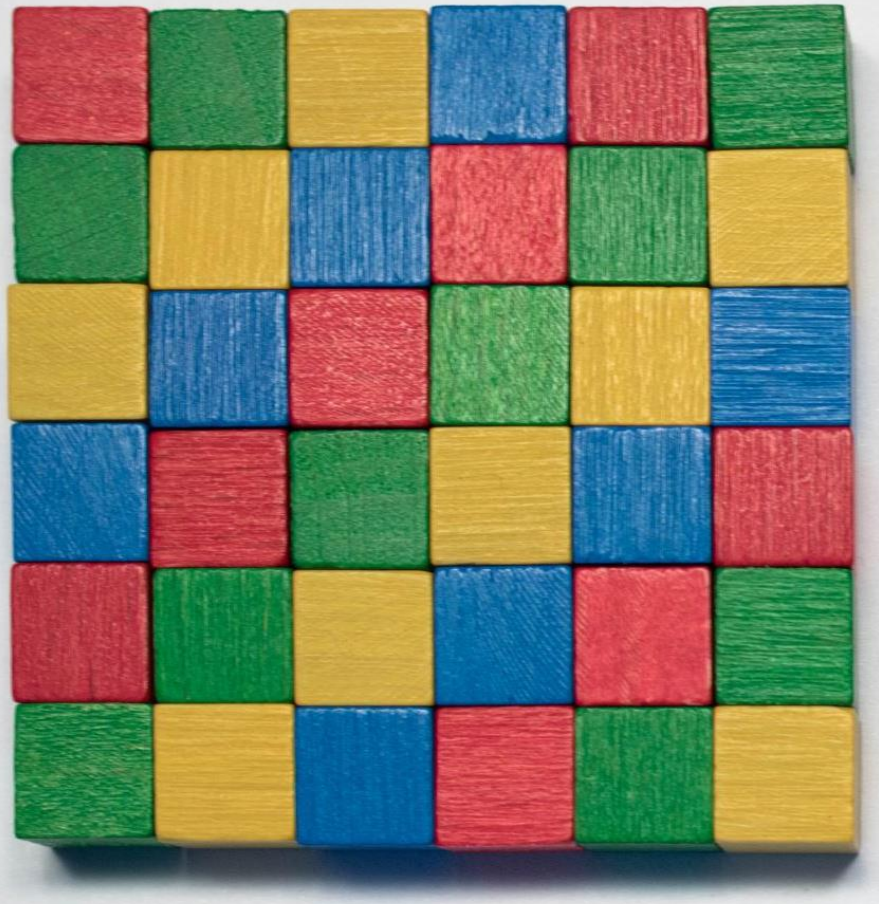
z-index provides solution when position overlay happens

Fixed Positioning

- For the most part, fixed positioning works just like absolute positioning. The significant difference is that the offset values for fixed elements are always relative to the viewport, which means the positioned element stays put even when the rest of the page scrolls.

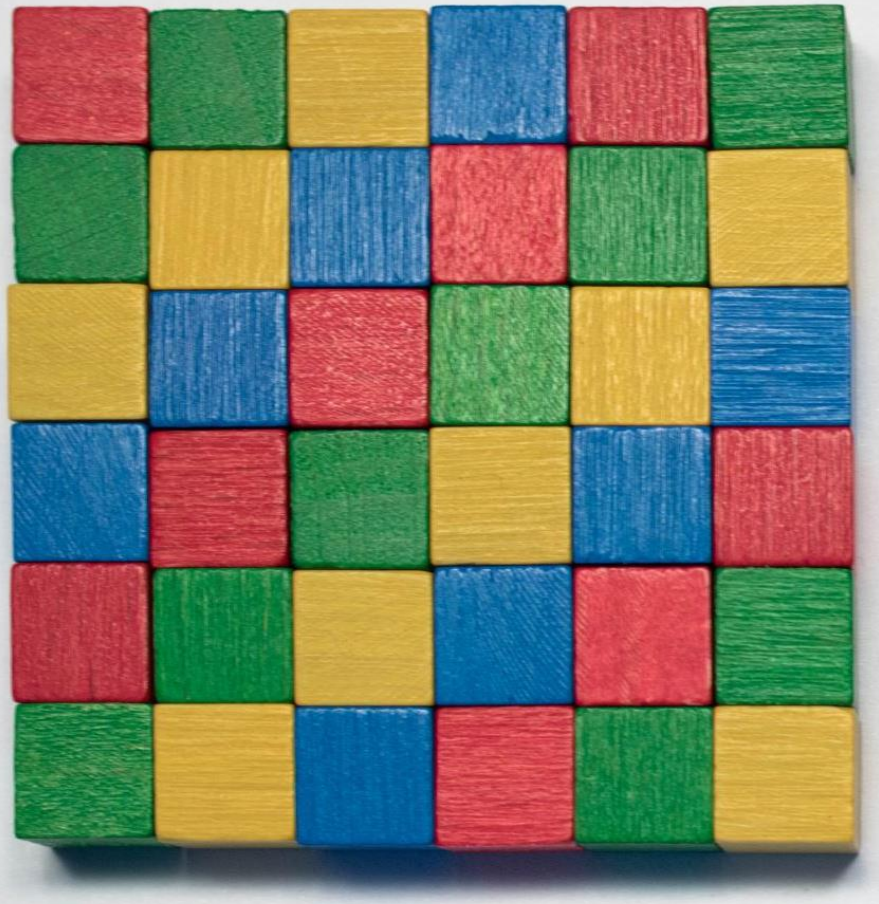
Masonry JavaScript Library (optional)

SECTION 11



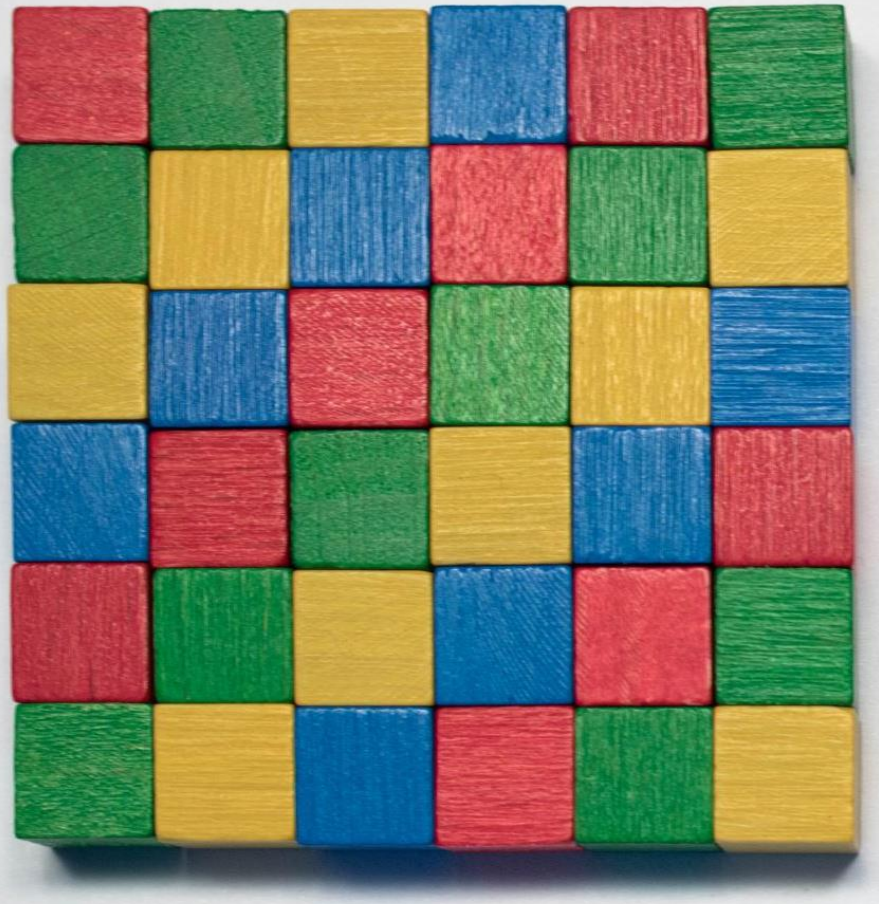
Masonry JavaScript Library

- In this case it would be really helpful to have a horizontal version of the default CSS floating. Instead of only arranging elements horizontally you also want to float them vertically, positioning each element in the next open spot in the grid.
- And that's when the **Masonry JavaScript library** comes into play.
- **Masonry**, developed by David DeSandro, is a JavaScript library, that places your selected elements in an optimal position based on available vertical space.
- As its name implies, it fits your elements in your grid like mason stones in a wall. As a result, vertical gaps between elements of varying heights get minimized.



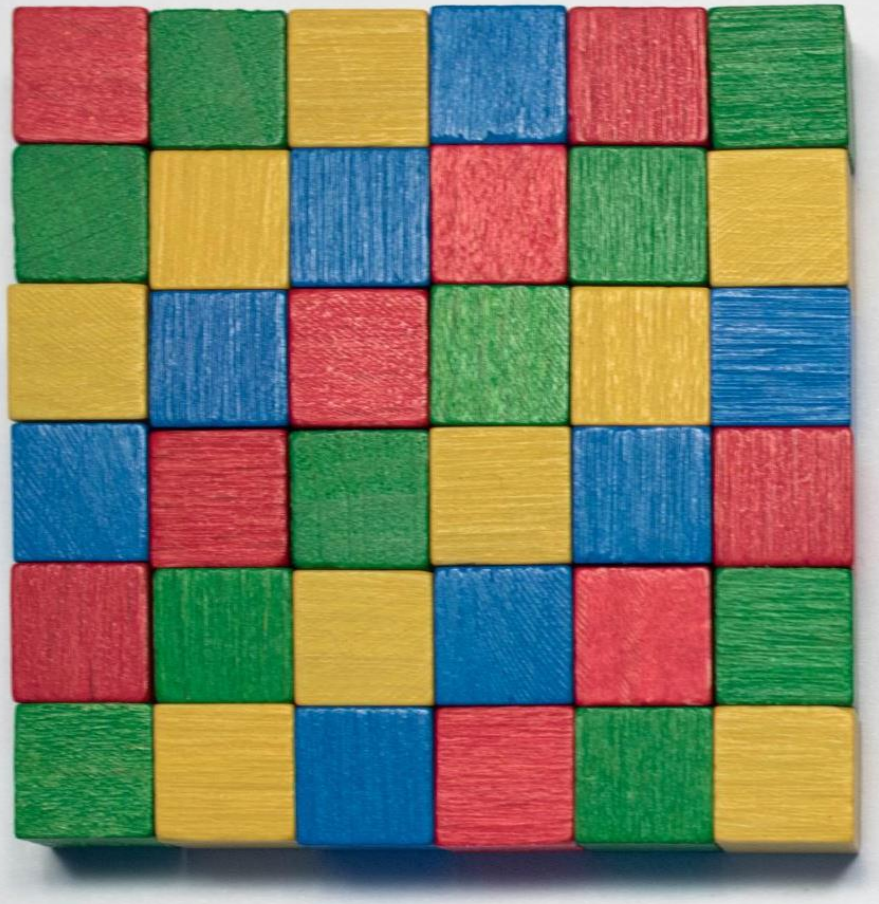
Masonry JavaScript Library

- To get started with Masonry you only need to **add a single JavaScript file** to your Drupal site. There are several ways to do so.
- I prefer to either add the script within the head section of my HTML template file by using the script tag.
- You can also add the script path to the .info file of your theme. In the following example, we will also use **jQuery** which comes with Drupal by default. However please note that jQuery is not required to use Masonry.



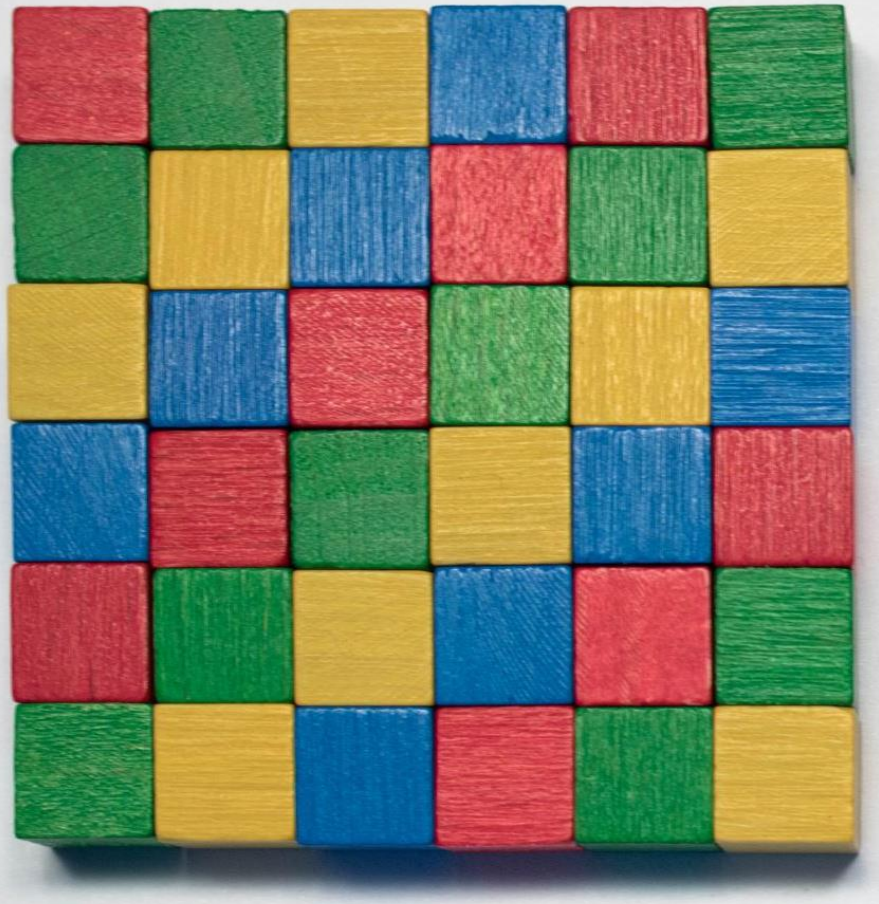
Masonry JavaScript Library

- Masonry works for every container element that has a group of similar child items. In the case of our Drupal view, the container element is the element inside the view with the class "view-content".
- The view rows with the class "views-row" are the child items. You can initialize the plugin either inline in your HTML by adding the class “js-masonry“ to the container element as well as setting the data-masonry-options attribute or with a few lines of JavaScript code.
- I strongly recommend separating markup and functionality on a website. That's why I prefer the JavaScript option.



Masonry JavaScript Library

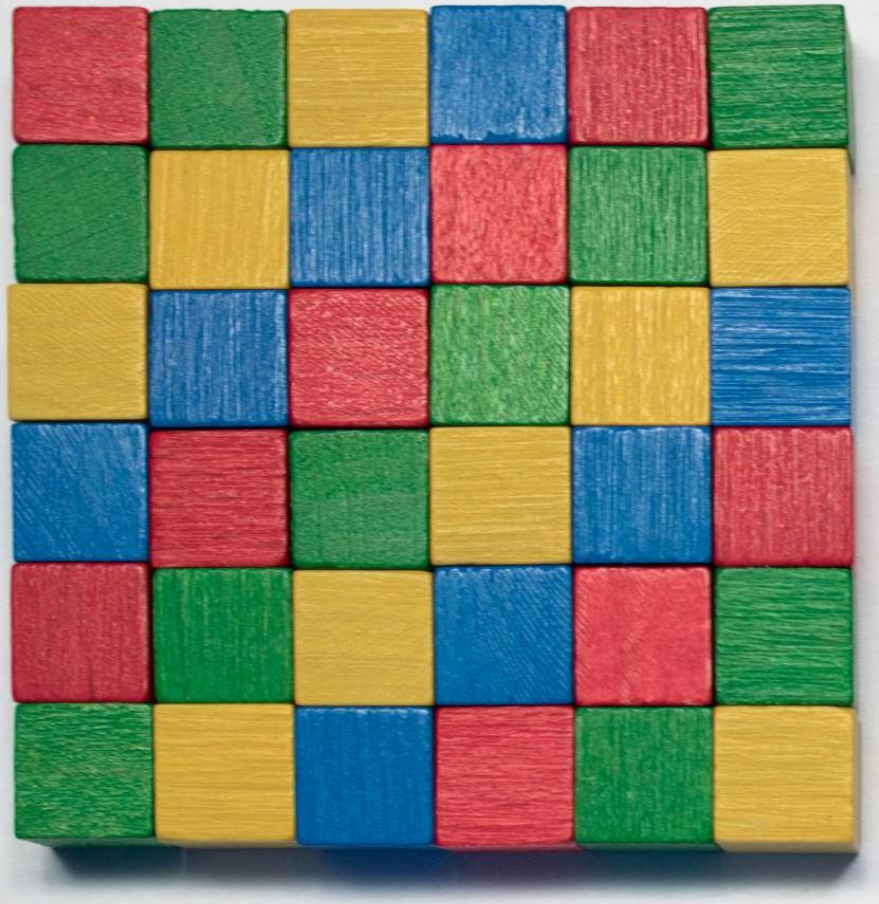
- To initialize Masonry you create a new Masonry instance in one of your JavaScript files which is also included in the Drupal site. The Masonry constructor accepts two arguments: The container element and an options object.
- The container element can be selected with jQuery and then passed to the function, along with a variety of non-required options. However setting the itemSelector option is always recommended to make sure the correct child elements are specified as item elements



Masonry JavaScript Library

- After selecting the container element and setting the **itemSelector** option your JavaScript should look like this:

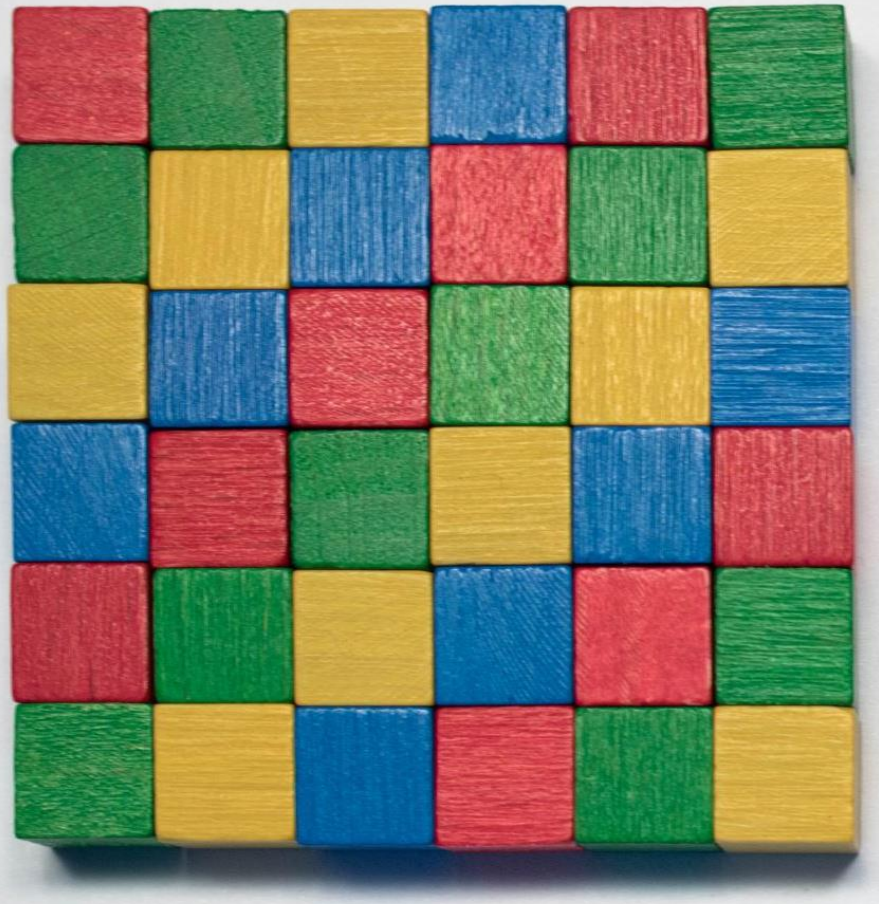
```
var container = $('.view-name.view-display-id-display_name .view-content');
var msnry = new Masonry( container, {
    // options
    itemSelector: '.views-row';
});
```



Masonry JavaScript Library

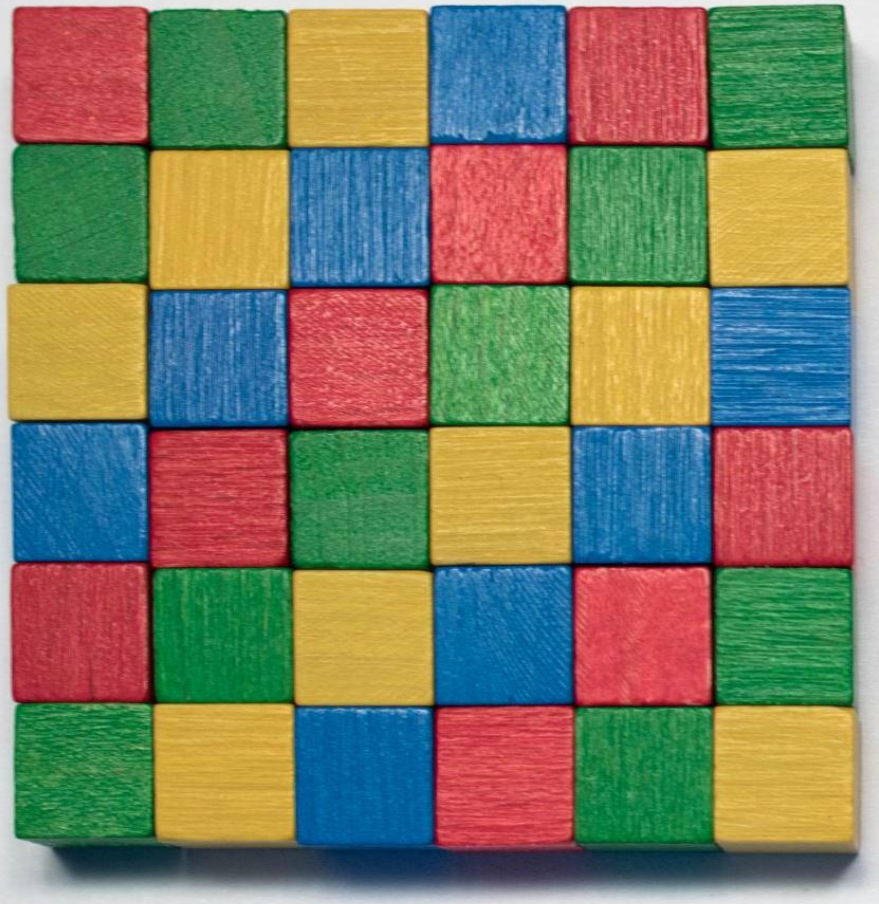
- You can minimize the lines of code by using **Masonry** as a jQuery plugin and applying it directly to the selected container element:

```
$('.view-name.view-display-id-display_name .view-content').masonry({
    itemSelector: '.views-row'
});
```



Masonry JavaScript Library

- There are several options you can define inside the options object as e.g. the column width of your grid and the gutter between the columns. As long as no column width is set, all sizing of your grid is handled by your CSS. The Masonry plugin acts responsively. It adjusts its positioning to the column width set in your CSS mediaqueries. If the options are not sufficient, there are also actions you can take on your masonry instance dynamically. Those methods allow you e.g. to append or prepend further elements to your grid, remove certain elements and redefine the layout of your grid in general.



Masonry JavaScript Library

- After studying all those methods it appears you can most anything you'd need to with Masonry. So next time you're working with grid layouts, surf over to http://masonry.desandro.com/ and give it a shot.

jQuery Masonry

SECTION 12

jQuery Masonry

- **jQuery Masonry** is the most popular solution to pull off this type of layout. It utilizes some pretty fancy JavaScript to reflow a series of divs.
- Putting Masonry into practice is pretty easy, all you need is a container that holds a series of divs that you want to arrange masonry style. You can place anything you want inside the **div**s, in this case I threw in some placeholder images.
- Once you have that in order, toss in jQuery and jQuery Masonry. Then you need to create a simple function that identifies your container and targets the class that we used for our Masonry images **div**s.

HTML

```
<div id="container">
    <div class="masonryImage">
        <img src="http://designshack.co.uk/wp-
content/uploads/http://lorempixum.com/200/200/food/1" alt="">
    </div>
    <div class="masonryImage">
        <img src="http://designshack.co.uk/wp-
content/uploads/http://lorempixum.com/200/150/food/10" alt="">
    </div>
    <div class="masonryImage">
        <img src="http://designshack.co.uk/wp-
content/uploads/http://lorempixum.com/200/250/food/3" alt="">
    </div>
    . . .
</div>
```

JavaScript

```
<script src="jquery-1.7.1.min.js"></script>
<script src="jquery.masonry.min.js"></script>
<script>
  $(function(){
    var $container = $('#container');
    $container.imagesLoaded( function(){
      $container.masonry({
        itemSelector : '.masonryImage'
      });
    });
  });
</script>
```

CSS

```
#container {
    width: 1200px;
    margin: 0 auto;
}
/*Media Queries*/
@media only screen and (max-width : 1199px),
only screen and (max-device-width : 1199px){
    #container {
        width: 1000px;
    }
}
@media only screen and (max-width : 999px),
only screen and (max-device-width : 999px){
    #container {
        width: 800px;
    }
}
```

CSS

```
@media only screen and (max-width : 799px),
only screen and (max-device-width : 799px){
    #container {
        width: 600px;
    }
}
@media only screen and (max-width : 599px),
only screen and (max-device-width : 599px){
    #container {
        width: 400px;
    }
}
@media only screen and (max-width : 399px),
only screen and (max-device-width : 399px){
    #container {
        width: 200px;
    }
}
```

Example

See It In Action

- This gives us a nice tight image gallery that perfectly responds to various browser window sizes. I've taken all the spacing out to illustrate just how tight you can get your images, but feel free to build in some margins so your images are spaced out a bit more.

http://designshack.net/tutorialexamples/masonry/demos/masonry.html