

More CSS

The contents and slides of this topic are used with permission from:

- Jennifer Robbins, Learning Web Design, O'Reilly, 5th edition, May 2018, ISBN 978-1-491-96020-2.
- Paul S. Wang, Dynamic Web programming and HTML5, Routledge, 1 edition, 2012, ISBN 1439871825.

CSS Colors

- ▶ CSS color names
- ▶ RGB and HSL color values
- ▶ Foreground and background colors
- ▶ Tiling background images
- ▶ More selectors and external style sheets

Named Color Values

Specify foreground or background color using one of 140 predefined CSS3 **color names**:

```
h1 { color: red; }  
h2 { color: darkviolet; }  
body { background-color: papayawhip; }
```

learningwebdesign.com/colornames.html

Numeric Color Values

For more control, define colors numerically using one of these color models:

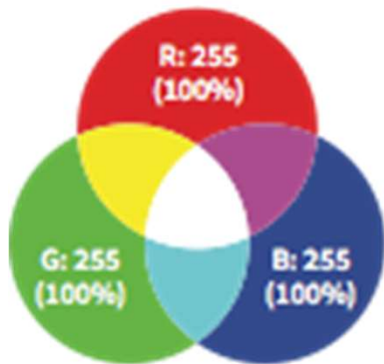
- ▶ **RGB** (combination of red, green, and blue light values)
- ▶ **RGBa** (RGB plus alpha transparency)
- ▶ **HSL** (hue, saturation, and luminosity)
- ▶ **HSLa** (HSL plus alpha transparency)

RGB Color

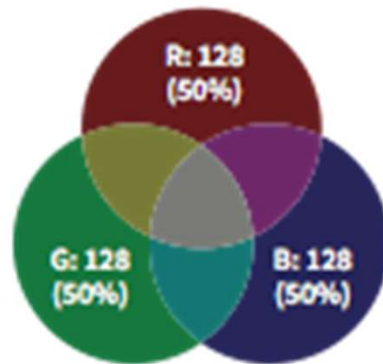
The **RGB color model** mixes color with red, green, and blue light.

Each channel can have 256 shades, for millions of color options.

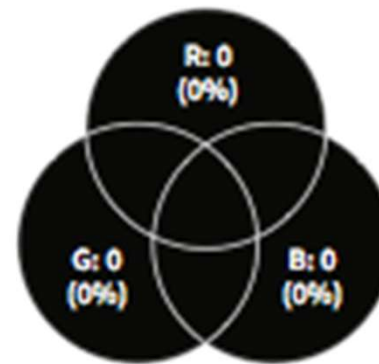
The RGB Color Model



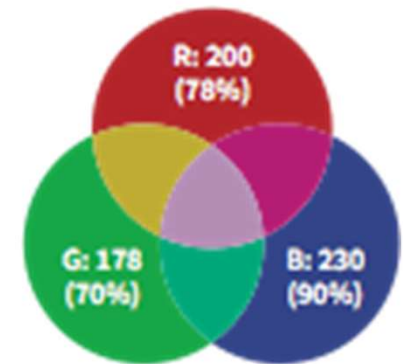
RGB: 255, 255, 255
white



RGB: 128, 128, 128
gray



RGB: 0, 0, 0
black



RGB: 200, 178, 230
pleasant lavender

RGB Values in Style Rules

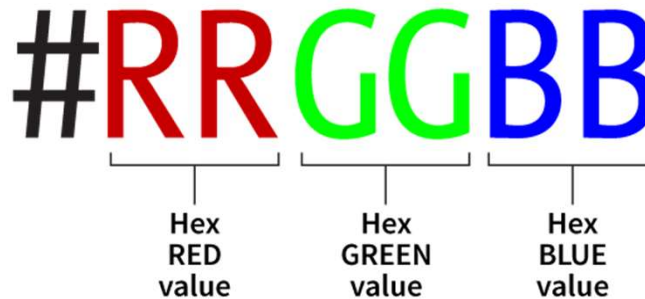
There are four formats for providing RGB color values:

- ▶ RGB values (0 to 255): `rgb(200, 178, 230)`
- ▶ Percentage values: `rgb(78%, 70%, 90%)`
- ▶ Hexadecimal values: `#C8B2E6`
- ▶ Condensed hexadecimal values (for double-digits only):
`#F06` is the same as `#FF0066`

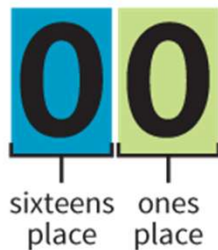
Hexadecimal RGB Values

Red, green, and blue values converted to hexadecimal and preceded by the # symbol.

Hexadecimal RGB values must be preceded by the # symbol.



Decimal	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Hex	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F



The decimal number **32** is represented as

20

2 sixteens and 0 ones

The decimal number **42** is represented as

2A

2 sixteens and 10 ones

RGBa Color

- ▶ **RGB + an alpha channel** for transparency
- ▶ The first three values are RGB. The fourth is the transparency level from 0 (transparent) to 1 (opaque).

Playing with RGBa

Playing with RGBa

Playing with RGBa

```
color: rgba(0, 0, 0, .1);
```

```
color: rgba(0, 0, 0, .5);
```

```
color: rgba(0, 0, 0, 1);
```


HSL and HSLa

- ▶ Colors described by values for **hue** ($^{\circ}$), **saturation** (%), and **luminosity** (%):

`hsl (180 , 50% , 75%)`

- ▶ Hue specifies the position on a color wheel (in degrees) that has red at 0° , green at 120° , and blue at 240° .
- ▶ HSL is less commonly used than RGB, but some find it more intuitive.
- ▶ HSLa adds an alpha value for transparency.

Foreground Color

color

Values: *Color value* (named or numeric)

Example: `blockquote {border: 4px dashed; color: green; }`

The **foreground** of an element consists of its text and border (if one is specified).

In the latitude of central New England, cabbages are not secure from injury from frost with less than a foot of earth thrown over the heads. In mild winters a covering of half that depth will be sufficient; but as we have no prophets to foretell our mild winters, a foot of earth is safer than six inches.

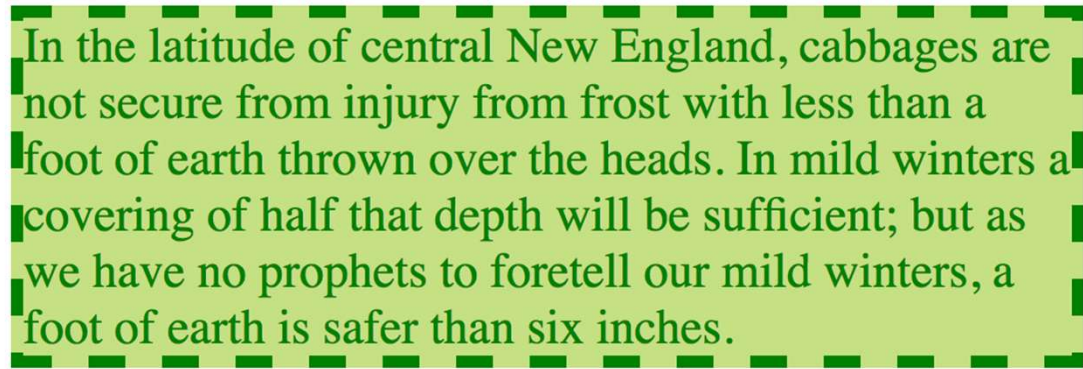
Background Color

`background-color`

Values: *Color value* (named or numeric)

Example: `blockquote {border: 4px dashed;
background-color: green;}`

The **background painting area** of an element fills the area behind the text to the outer edge of the border.



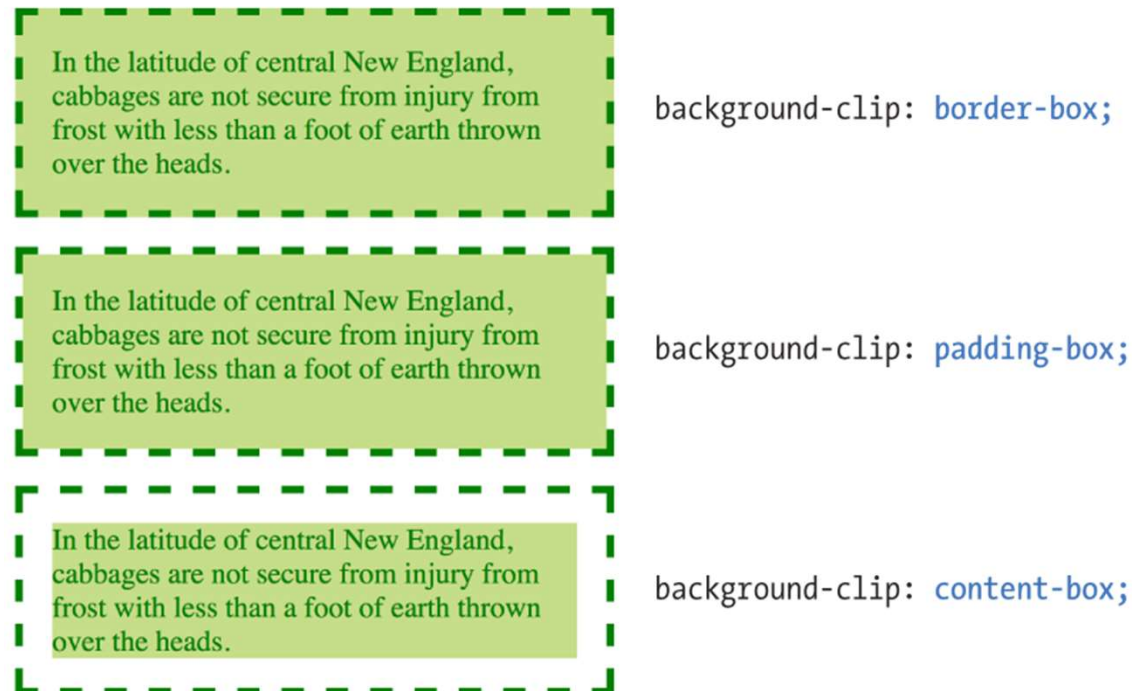
In the latitude of central New England, cabbages are not secure from injury from frost with less than a foot of earth thrown over the heads. In mild winters a covering of half that depth will be sufficient; but as we have no prophets to foretell our mild winters, a foot of earth is safer than six inches.

Clipping the Background

`background-clip`

Values: `border-box`, `padding-box`, `content-box`

Specifies where the background painting area ends.



Opacity

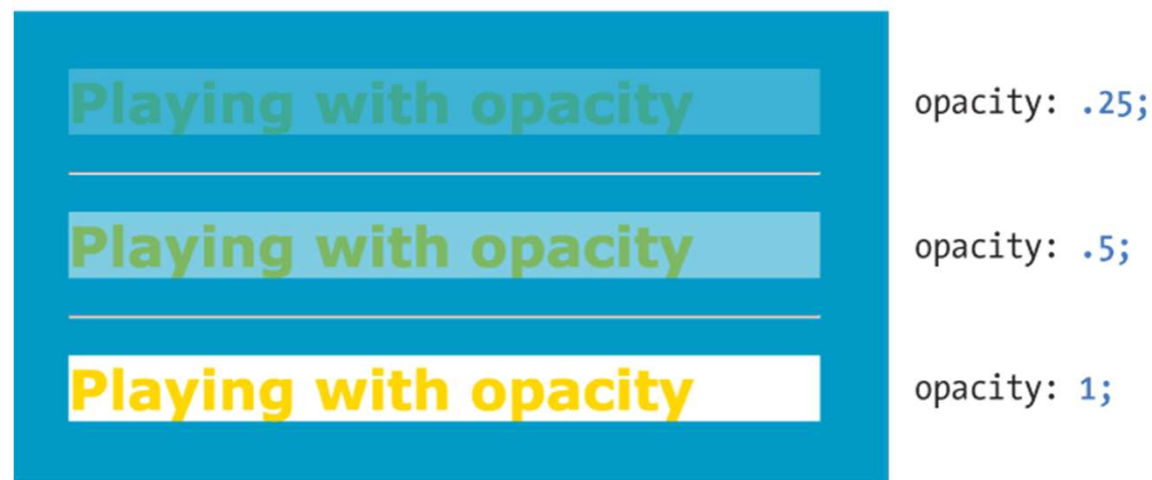
`opacity`

Values: *number* (0 to 1)

Example:

```
h1 {color: gold; background: white; opacity: .25;}
```

Specifies the transparency level from 0 (transparent) to 1 (opaque):



Tiling Background Images

`background-image`

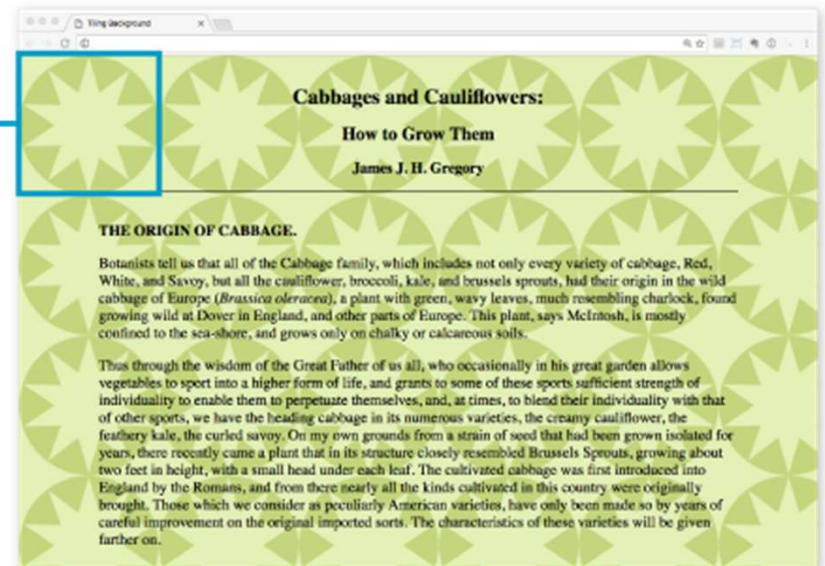
Values: `url (location of image)`, `none`

Example: `body {background-image: url(star.png);}`

Locates an image to be used as a tiling background image behind an element. By default, it starts at the top, left corner and repeats horizontally and vertically:



`star.png`
150 × 150 pixels



Background Repeating

`background-repeat`

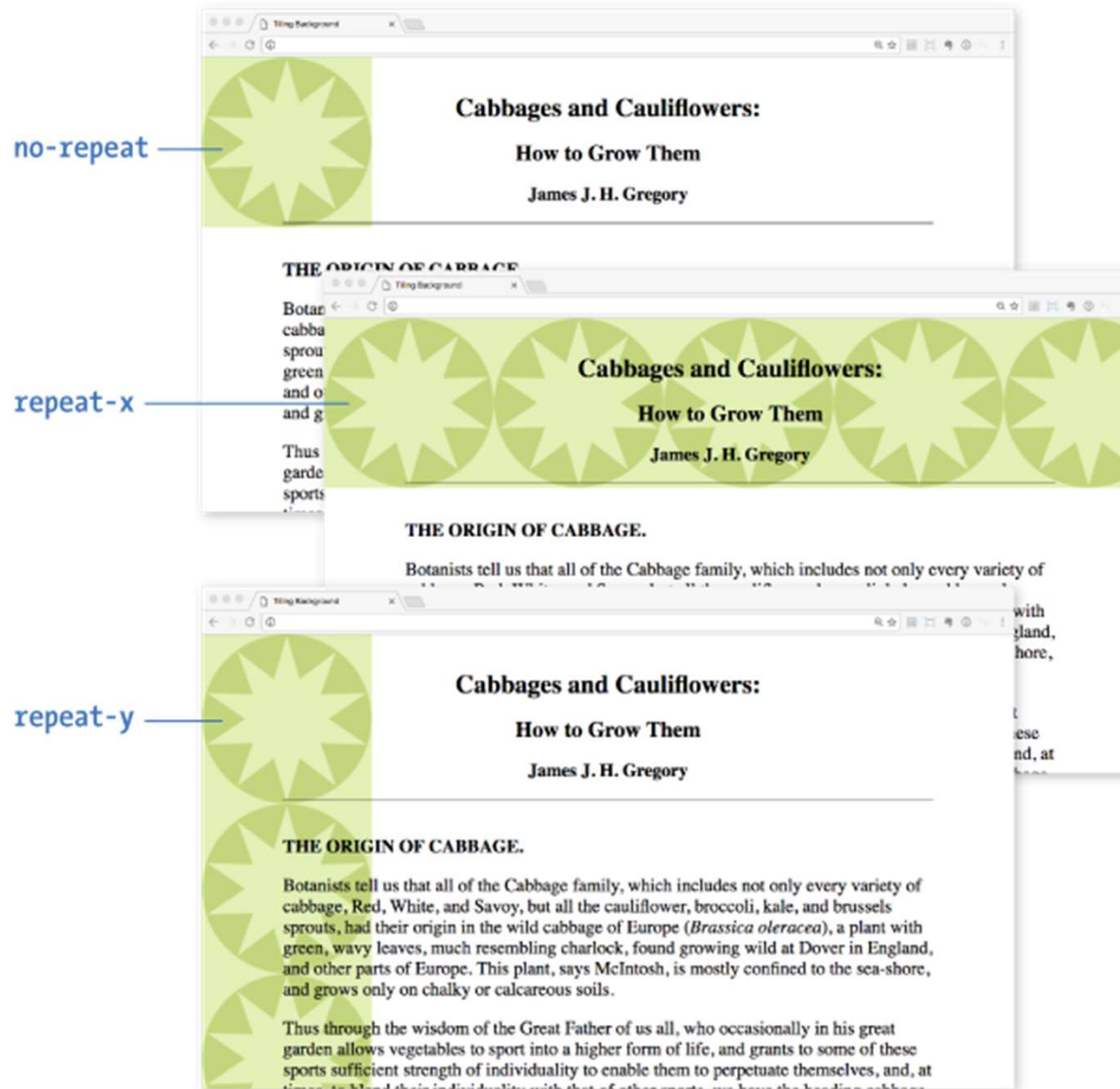
Values:

`repeat`, `no-repeat`, `repeat-x`, `repeat-y`, `space`, `round`

Specifies how the background image repeats and can restrict it to tiling in one direction or not at all:

- ▶ `repeat-x`: Tiles horizontally only
- ▶ `repeat-y`: Tiles vertically only
- ▶ `space`: Adds space around images so they fit in the window with no clipping
- ▶ `round`: Distorts the image so it fits without clipping

Background Repeating (cont'd)



Background Position

`background-position`

Values:

Length, percentage, left, center, right, top, bottom

Specifies the position of the **origin image**, the first image that is placed in the background from which tiling images extend.

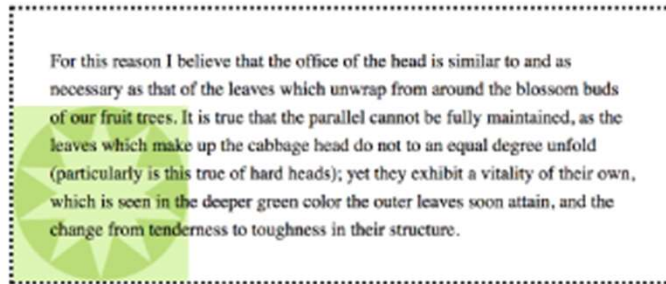
Examples (horizontal position goes first):

```
background-position: left bottom;
```

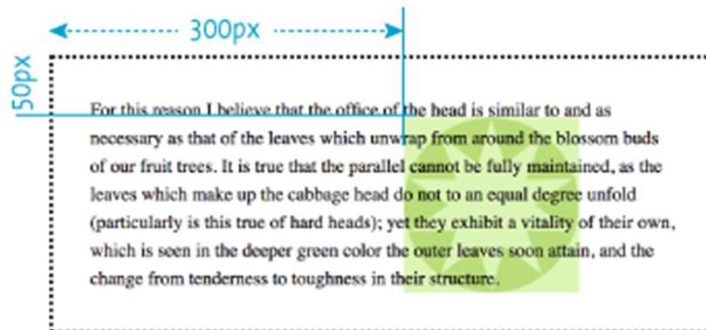
```
background-position: 300px 100px;
```

```
background-position: 25% 100%;
```

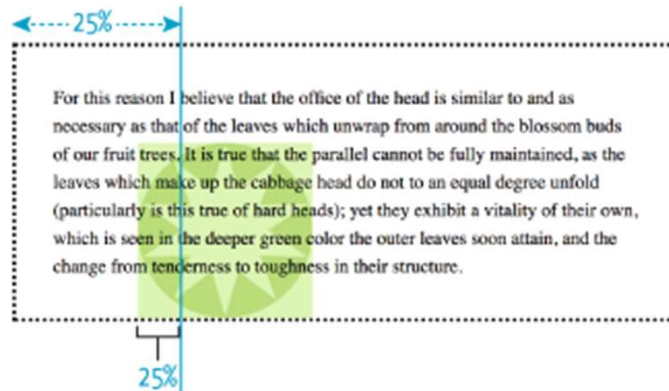
Background Position (cont'd)



background-position: left bottom;



background-position: 300px 50px;



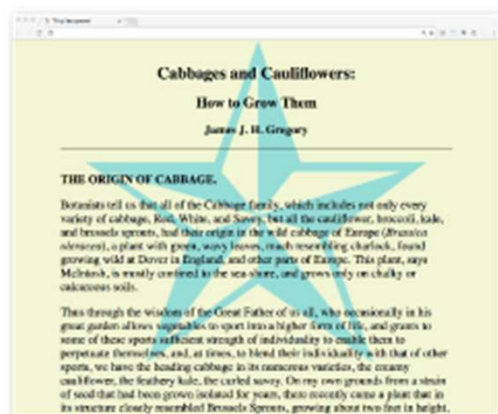
background-position: 25% 100%;

Background Attachment

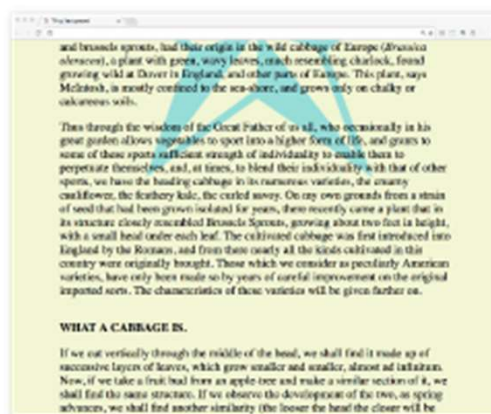
background-attachment

Values: scroll, fixed, local

Specifies whether the background image scrolls with the content or stays in a fixed position relative to the viewport.



A large non-repeating background image in the body of the document.



background-attachment: scroll;

By default, the background image is attached to the body element and scrolls off the page when the page content scrolls.



background-attachment: fixed;

When background-attachment is set to fixed, the image stays in its position relative to the browser viewing area and does not scroll with the content.

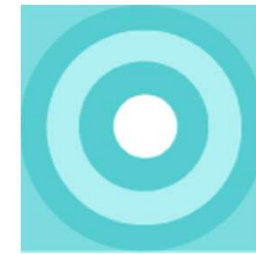
Background Size

background-size

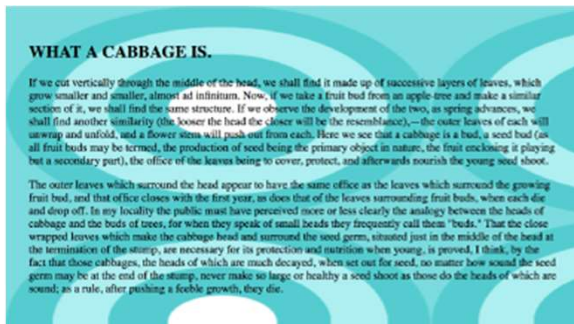
Values:

Length, percentage, auto, cover, contain

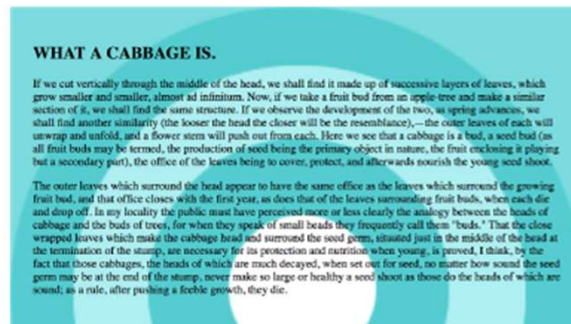
Specifies the size of the tiling image:



target.png
300 × 300 pixels

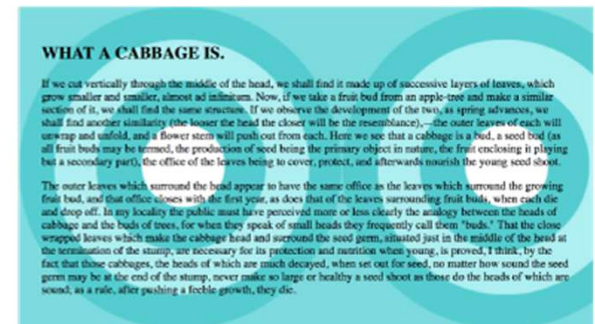


background-size: 600px 300px;



background-size: cover;

The entire background area of the element is covered, and the image maintains its proportions even if it is clipped.



background-size: contain;

The image is sized proportionally so it fits entirely in the element. There may be room left over for tiling (as shown).

Shorthand background Property

background

Values:

*background-color background-image background-repeat
background-attachment background-position background-clip
background-origin background-size*

- ▶ Specifies all background properties in one declaration

```
background: white url(star.png) no-repeat top  
center fixed;
```

- ▶ Properties are optional and may appear in any order
- ▶ Properties not represented reset to their defaults—**be careful it doesn't overwrite previous background settings.**

Multiple Background Images

You can place more than one background image in a single image (separated by commas):

```
body {  
  background:  
    url(image1.png) left top no-repeat,  
    url(image2.png) center center no-repeat,  
    url(image3.png) right bottom no-repeat;  
}
```

Gradient Fills

- ▶ A **gradient** is a transition from one color to another.
- ▶ **Linear gradients** change colors along a line.
- ▶ **Radial gradients** start at a point and spread outward in a circular or elliptical shape.
- ▶ You can generate a gradient image for use as a background using `linear-gradient()` and `radial-gradient()` notation.

Example:

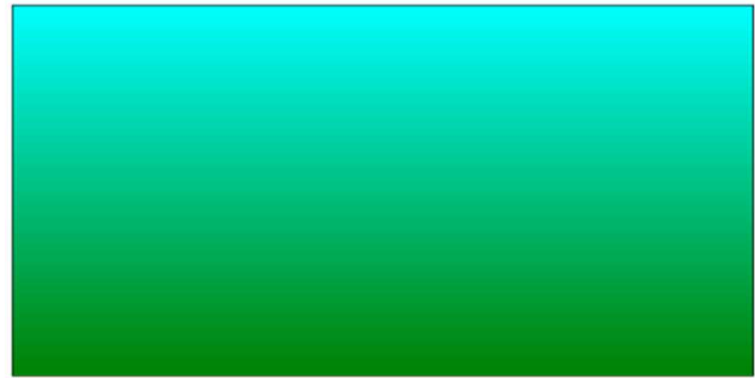
```
#banner {  
    background-image: linear-gradient(180deg, aqua,  
green) ;  
}
```

Linear Gradient

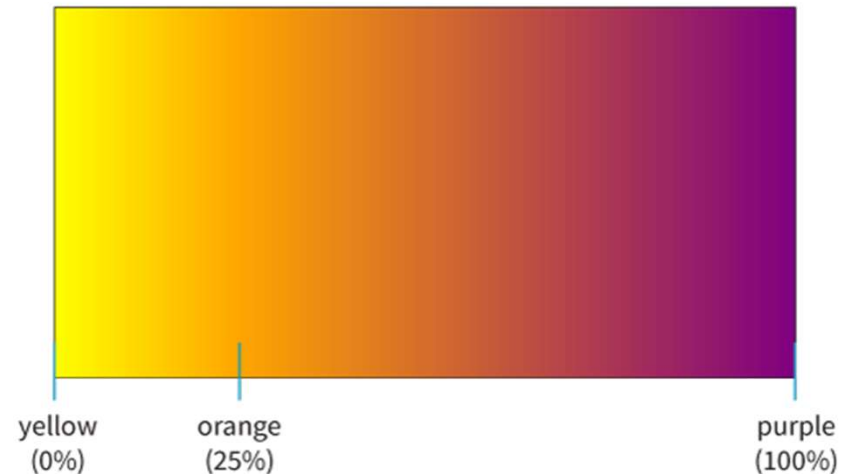
The `linear-gradient()` notation provides the angle of a gradient line and the colors the line passes through.

It is specified in degrees (`deg`) or keywords (`to top`, `to right`, `to bottom`, `to left`).

```
linear-gradient(180deg, aqua, green);  
or  
linear-gradient(to bottom, aqua, green);
```



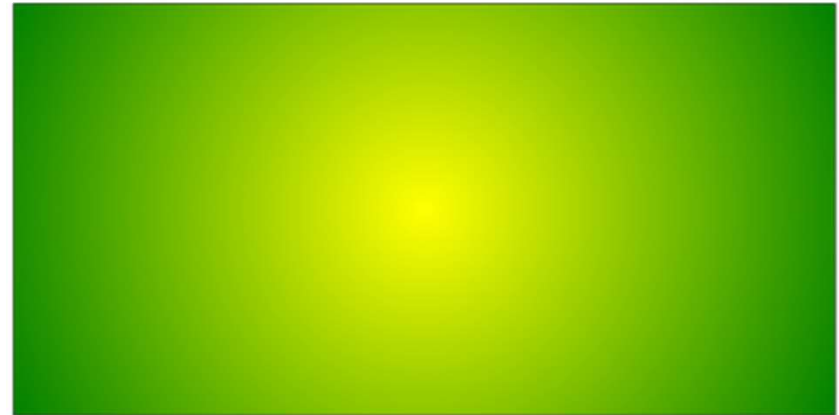
```
linear-gradient(90deg, yellow, orange 25%, purple);
```



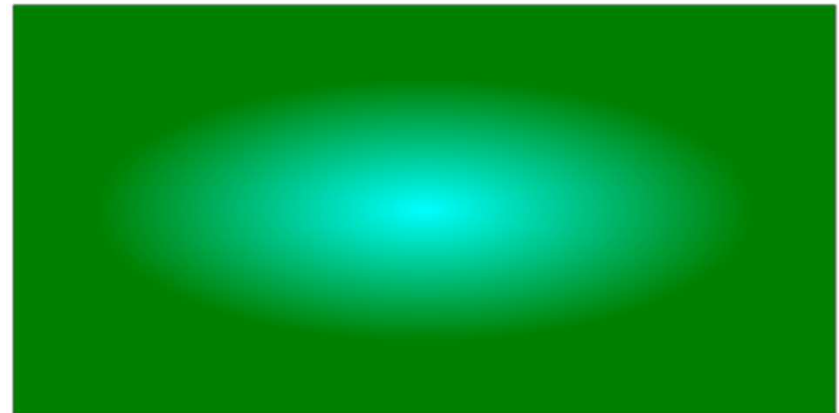
Radial Gradients

The `radial-gradient()` notation provides the color values and optional size, shape, and position information.

```
radial-gradient(yellow, green);
```



```
radial-gradient(200px 80px, aqua, green);
```



More Selector Types

Pseudo-class selectors

Pseudo-element selectors

Attribute selectors

Pseudo-Class Selectors

Treat elements in a certain state as belonging to the same class

Link Pseudo-classes

:link Applies style to unvisited (unclicked) links

:visited Applies style to visited links

User Action Pseudo-classes

:focus Applies when element is selected for input

:hover Applies when the mouse pointer is over the element

:active Applies when the element (such as a link or button) is in the process of being clicked or tapped

Pseudo-classes (cont'd)

Pseudo-classes must appear in the following order:

```
a { text-decoration: none; } /* turns underlines off for all links */
a:link { color: maroon; }
a:visited { color: gray; }
a:focus { color: maroon; background-color: #ffd9d9; }
a:hover { color: maroon; background-color: #ffd9d9; }
a:active { color: red; background-color: #ffd9d9; }
```

Samples of my work:

- Pen and Ink Illustrations
- Paintings
- Collage

`a:link`

Links are maroon and not underlined.

Samples of my work:

- Pen and Ink Illustrations
- Paintings
- Collage

`a:focus`
`a:hover`

While the mouse is over the link or when the link has focus, the pink background color appears.

Samples of my work:

- Pen and Ink Illustrations
- Paintings
- Collage

`a:active`

As the mouse button is being pressed, the link turns bright red.

Samples of my work:

- Pen and Ink Illustrations
- Paintings
- Collage

`a:visited`

After that link has been visited, the link is gray.

Pseudo-Element Selectors

Applies styles to elements not explicitly marked up in the source.

::first-line

Applies a style to the first line of an element:

```
p:first-line {letter-spacing: 9px;}
```

::first-letter

Applies a style to the first letter of an element:

```
p:first-letter { font-size 300%; color: orange; }
```

Generated Content Example



We are required to warn you that undercooked food is a health risk. **Thank you.**

The style sheet:

```
p.warning::before {
  content: url(exclamation.png);
  margin-right: 6px;
}
p.warning::after {
  content: " Thank you.";
  color: red;
}
```

The markup:

```
<p class="warning">We are required to warn you that
undercooked food is a health risk.</p>
```

Attribute Selectors

Targets elements based on attribute names or values. There are eight types:

Simple attribute selector

Matches an element with a given attribute:

E[*attribute*]

```
img[title] { border: 3px solid; }
```

(Matches every `img` element that has a `title` attribute)

Attribute Selectors (cont'd)

Exact attribute value selector

Matches an element with a specific value for an attribute:

`E[attribute="exact value"]`

```
img[title="first grade"] {border: 3px solid;}
```

(matches only if the title value is "first grade")

Partial attribute value selector (~)

Matches an element by one part of an attribute value.:

`E[attribute~="value"]`

```
img[title~="grade"] {border: 3px solid;}
```

(matches "first grade", "second grade", and so on)

Attribute Selectors (cont'd.)

Beginning substring attribute value selector (^)

Matches an element with attribute values that start with the given string of characters:

```
E[attribute^="first part of a value"]
```

Ending substring attribute value selector (\$)

Matches an element with attribute values that end with the given string of characters:

```
E[attribute$="last part of a value"]
```

Arbitrary substring attribute value selector (*)

Looks for the text string in any part of the attribute value:

```
E[attribute*="any part of the value"]
```

External Style Sheets

- ▶ Store styles in a separate `.css` file and attach to the document via `<link>` or `@import`
- ▶ Most efficient method: Change styles in multiple documents by editing one `.css` file
- ▶ A `.css` document is a simple text document (may begin with `@charset` to identify character set)

Attaching a Style Sheet with the link Element

- ▶ The `link` element defines a relationship between the current document and an external resource.
- ▶ It goes in the `head` of the document.
- ▶ Use the `rel` attribute to say it's a style sheet. Use `href` to provide the URL of the `.css` file (relative to the current document):

```
<head>  
  <title>Titles are required.</title>  
  <link rel="stylesheet"  
href="/path/styleSheet.css">  
</head>
```

Attaching a Style Sheet with an @import rule

An `@import` rule imports the contents of an external style sheet into another style sheet (either a `.css` document or embedded with `style`):

```
<head>
  <style>
    @import url("/path/styleSheet.css");
    p { font-face: Verdana;}
  </style>
  <title>Titles are required.</title>
</head>
```

Exercises

EXERCISE 13-7. Multiple background images

In this exercise, we'll give multiple background images a try (be sure you aren't using an old version of IE, or this won't work).

I'd like the dot pattern in the **header** to run along the left and right sides. I also have a little goose silhouette (*gooseshadow.png*) that might look cute walking along the bottom of the header. I'm making this example friendly for non-supporting browsers (IE8 and earlier) by providing a fallback declaration with just one image and separating out the **background-color** declaration so it doesn't get overridden. If IE8 is not a concern, you don't need the fallback.

You can see in the example that we are placing three images in a single header: dots on the left side, dots on the right, and a goose at the bottom.

```
header {  
  --  
  background: url(images/purpledotted.png) center top  
  repeat-x;  
  background:  
    url(images/purpledotted.png) left top repeat-y,  
    url(images/purpledotted.png) right top repeat-y,  
    url(images/gooseshadow.png) 90% bottom no-repeat;  
  background-color: rgba(255,255,255,.5);  
}
```

FIGURE 13-30 shows the final result. Meh, I liked it better before, but you get the idea.



FIGURE 13-30. The bistro menu header with two rows of dots and a small goose graphic in the **header** element.

EXERCISE 13-8. Making an external style sheet

It is OK to use an embedded style sheet while designing a page, but it is probably best moved to an external style sheet once the design is finished so it can be reused by multiple documents in the site. We'll do just that for the summer menu style sheet.

1. Open the latest version of *summer-menu.html*. Select and cut all of the rules within the **style** element, but leave the `<style>...</style>` tags because we'll be using them in a moment.
2. Create a new plain ASCII text document and paste all of the style rules. Make sure that no markup got in there by accident.
3. Save this document as *menustyles.css* in the same directory as the *summer-menu.html* document.
4. Now, back in *summer-menu.html*, add an **@import** rule to attach the external style sheet:

```
<style>  
  @import url(menustyles.css);  
</style>
```

Save the file and reload it in the browser. It should look exactly the same as it did when the style sheet was embedded. If not, go back and make sure that everything matches the examples.

5. Delete the whole **style** element, and this time we'll add the style sheet with a **link** element in the **head** of the document.

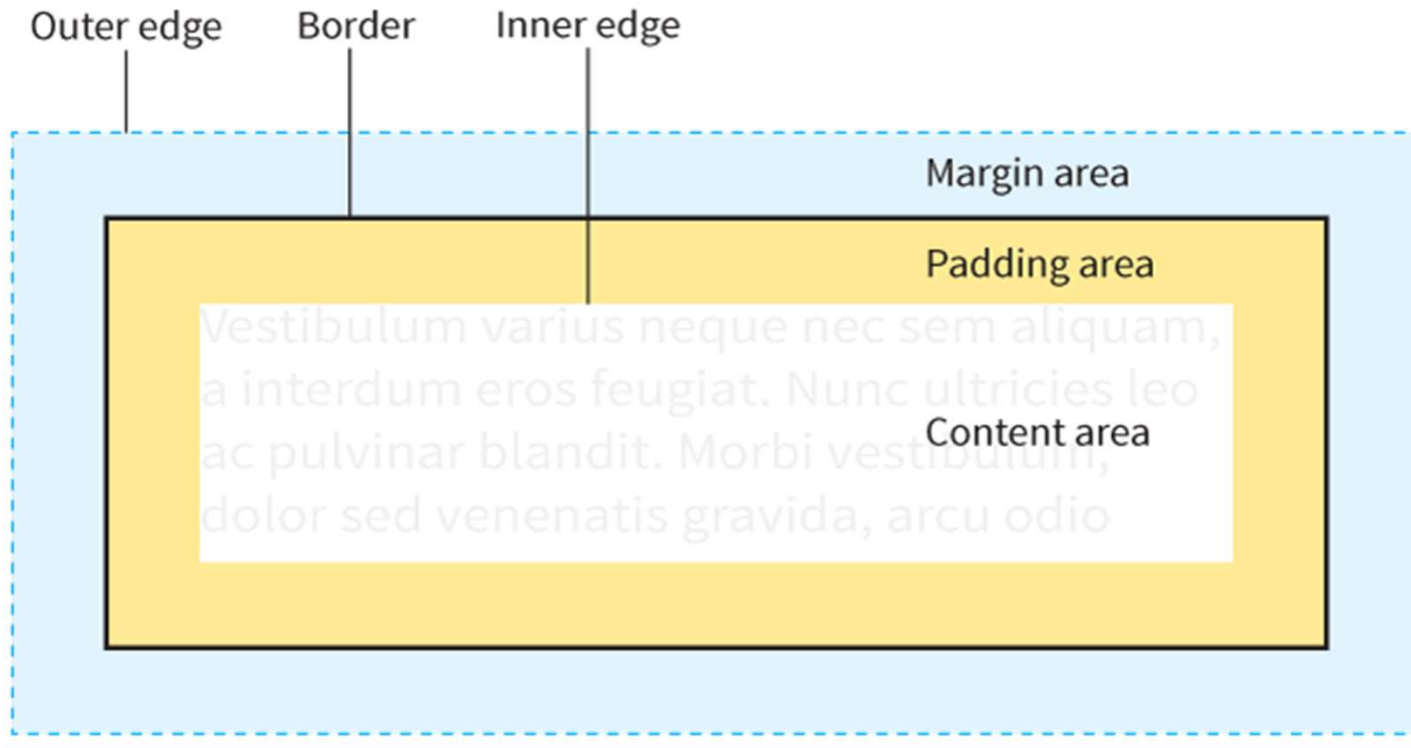
```
<link rel="stylesheet"  
  href="menustyles.css">
```

Again, test your work by saving the document and taking a look at it in the browser.

Thinking inside the box

- ▶ The parts of an element box
- ▶ Box dimensions
- ▶ Padding
- ▶ Borders
- ▶ Outlines
- ▶ Margins
- ▶ Display roles
- ▶ Drop shadows

The Parts of an Element Box



NOTE: The margin is indicated with a blue shade and outline, but is invisible in the layout.

Specifying Box Dimensions

width

Values: *Length, percentage*, auto

height

Values: *Length, percentage*, auto

Specify the dimensions of an element box with **width** and **height** properties

Box Sizing Models

`box-sizing`

Values: `content-box`, `border-box`

There are two methods for sizing an element box, specified with the `box-sizing` attribute:

Content-box sizing (default)

Applies `width` and `height` values to the content box only

Border-box sizing

Applies `width` and `height` values to the border box (including the padding and content)

Box Sizing Models Compared

← Total visible box width = 550px →
(50px of padding and 10px of border are added to 500px content box width)

```
box-sizing: content-box;  
width: 500px;
```

This week I am *extremely* excited about a new cooking technique called *sous vide*. In *sous vide* cooking, you submerge the food (usually vacuum-sealed in plastic) into a water bath that is precisely set to the target temperature you vacuum-sealed in plastic) into a water bath that is precisely set to the target temperature you want the food to be cooked to.

```
box-sizing: border-box;  
width: 500px;
```

← Total visible box width = 500px →
(50px of padding and 10px of border are included in the border-box size)

Overflow

overflow

Values: `visible`, `hidden`, `scroll`, `auto`

Specifies what to do when content doesn't fit in a sized element box:

visible

Applying the masks to the glasses is the most labor-intensive part of the process. Not only do you have to measure, place, and burnish on each mask, but you also need to completely cover the remainder of the glass in heavy paper. Any exposed areas (even inside) will get scratched by the flying sand, so it has to be a good seal.

hidden

Applying the masks to the glasses is the most labor-intensive part of the process. Not only do you have to measure, place, and burnish on each mask, but you also need to completely cover the remainder of the glass

scroll

labor-intensive part of the process. Not only do you have to measure, place, and burnish on each mask, but you also need to completely cover the remainder of the glass in heavy paper. Any exposed areas (even

auto (short text)

Applying the masks to the glasses is the most labor-intensive part of the process.

auto (long text)

Applying the masks to the glasses is the most labor-intensive part of the process. Not only do you have to measure, place, and burnish on each mask, but you also need to completely cover the remainder of the glass

Padding

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

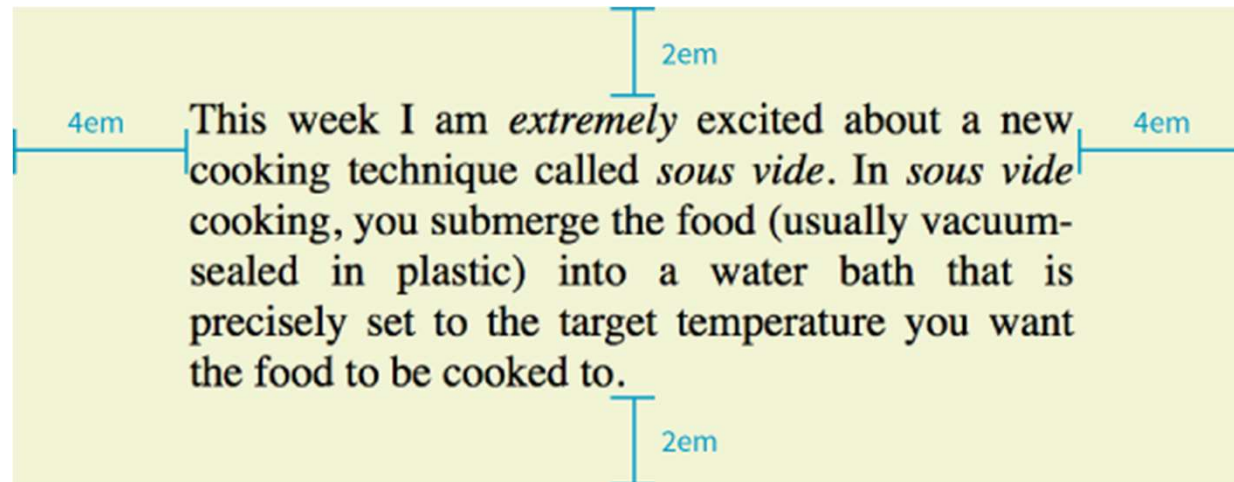
Values: *Length*, *percentage*

An amount of space between the content area and the border (or the space the border would be if one isn't specified).

You can add padding to one side at a time, or on all sides with the `padding` shorthand property.

Padding (cont'd)

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



Shorthand padding Property

The `padding` property adds space around 1, 2, 3, or 4 sides of the content using the clockwise top, right, bottom, left (TRouBLE) order:

```
padding: top right bottom left;
```

```
padding: 2em 4em 2em 4em;
```

(this shorthand produces the same result as the example on the previous slide)

Shorthand padding Property (cont'd)

If the left and right sides are the same, you can omit the last value, and the second value will be applied on both the left and right sides:

```
padding: top right+left bottom;
```

```
padding: 2em 4em 2em;
```

(this shorthand produces the same result as the examples on the two previous slides)

Shorthand padding Property (cont'd)

If the top and bottom sides are also the same, you can omit the third value, and the first value will be applied on both the top and bottom:

```
padding: top+bottom right+left;
```

```
padding: 2em 4em;
```

(same result as previous examples)

If all sides are the same, provide one value, and it's applied to all sides:

```
padding: all sides;
```

```
padding: 2em;
```

(2em padding all around)

Borders

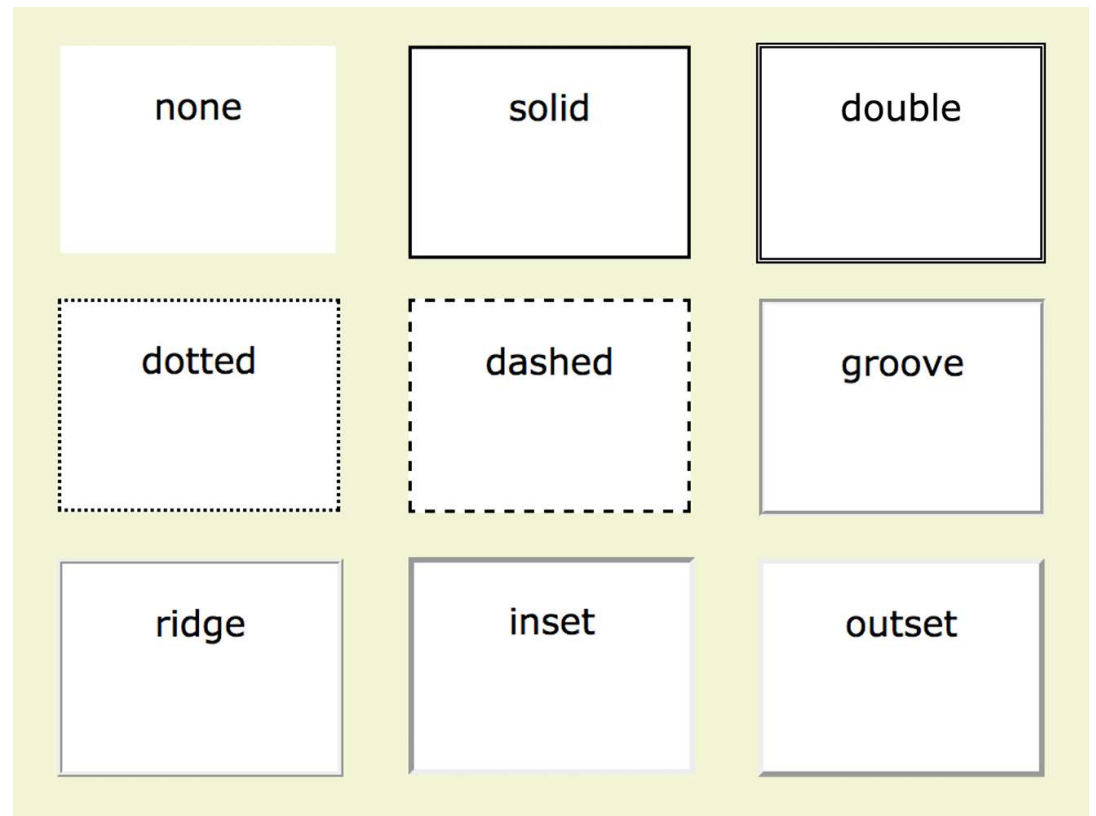
- ▶ A **border** is a line drawn around the content area and its (optional) padding.
- ▶ The thickness of the border is included in the dimensions of the element box.
- ▶ You define **style**, **width** (thickness), and **color** for borders.
- ▶ Borders can be applied to single sides or all around

Border Style

`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

NOTE: The default is none, so if you don't define a border style, it won't appear.

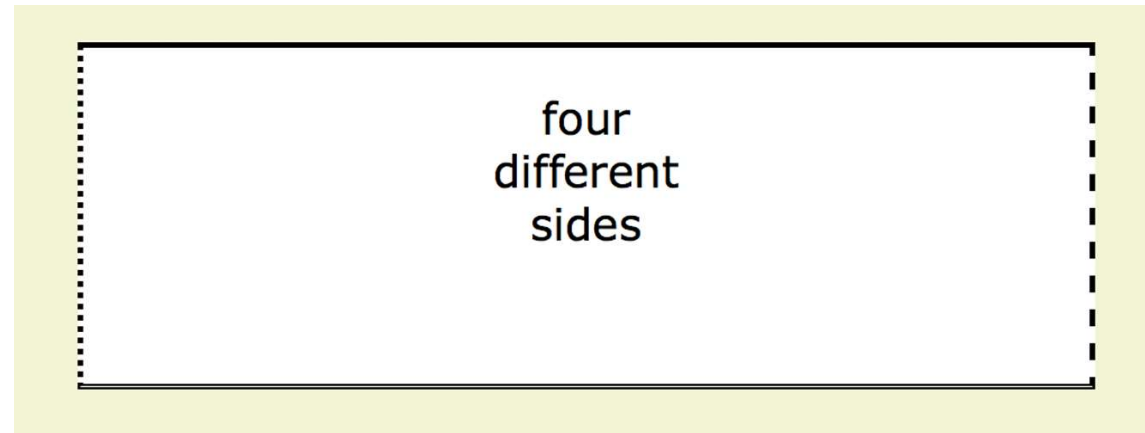


Border Style

`border-style`

The `border-style` shorthand uses the clockwise (TRouBLe) shorthand order. The following rules have the same effect:

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



Border Width

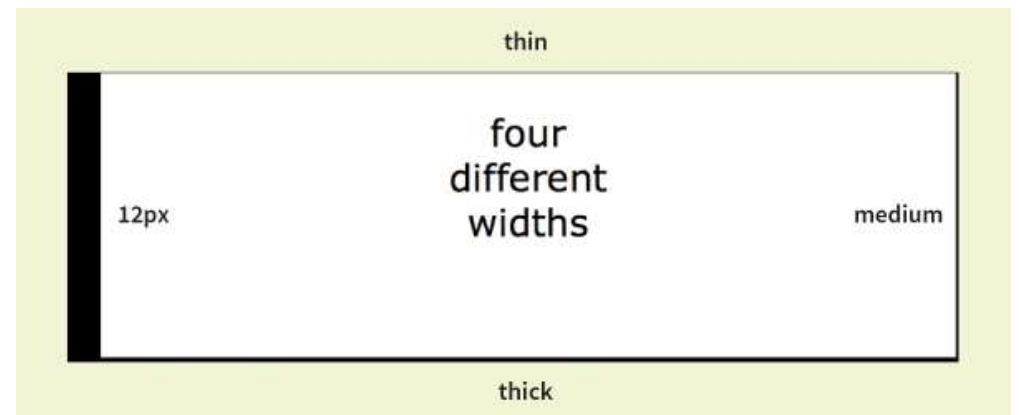
`border-width`,
`border-top-width`, `border-right-width`,
`border-bottom-width`, `border-left-width`

Values: *Length*, thin, medium, thick

The `border-width` shorthand uses the clockwise (TRouBLe) order:

```
div#help {  
  border-width: thin medium thick 12px;  
  border-style: solid;  
  width: 300px;  
  height: 100px;  
}
```

NOTE: The `border-style` property is required for the border to be rendered.



Border Color

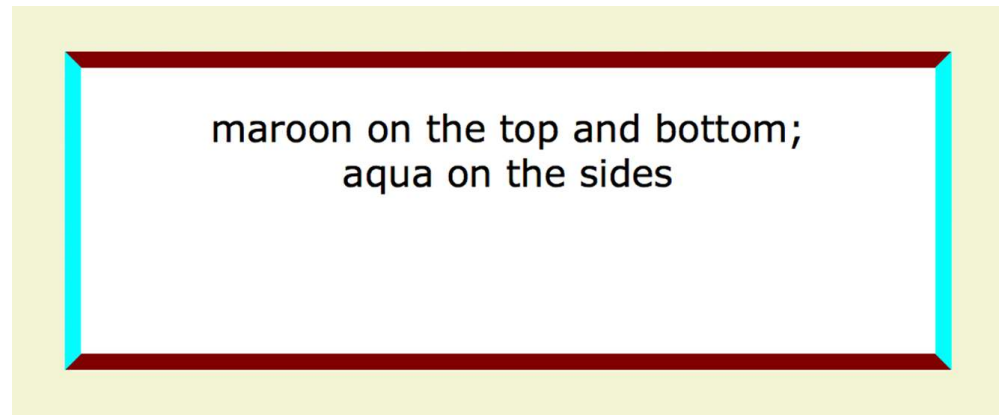
`border-color`,
`border-top-color`, `border-right-color`,
`border-bottom-color`, `border-left-color`

Values: *Color value (named or numeric)*

The `border-color` properties override the `color` property:

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

NOTE: The `border-style` property is required for the border to be rendered.



Border Shorthand Properties

`border`,
`border-top`, `border-right`,
`border-bottom`, `border-left`

Values: *border-style border-width border-color*

Combine style, width, and color values in shorthand properties for each side or all around (**border**):

```
p.example {  
  border: 2px dotted aqua;  
}
```

NOTE: The `border-style` property must be included in the shorthand for the border to be rendered.

Border Radius (Rounded Corners)

`border-radius`

Values: 1, 2, 3, or 4 length or percentage values

- The `border-radius` property rounds off the corners of an element.
- The value is a length or percentage value reflecting the radius of the curve.
- Providing one value makes all the corners the same.
- Four values are applied clockwise, starting from the top-left corner.

Border Radius (cont'd)

```
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;`

or

`border-radius: 1em 2em;`

Margins

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

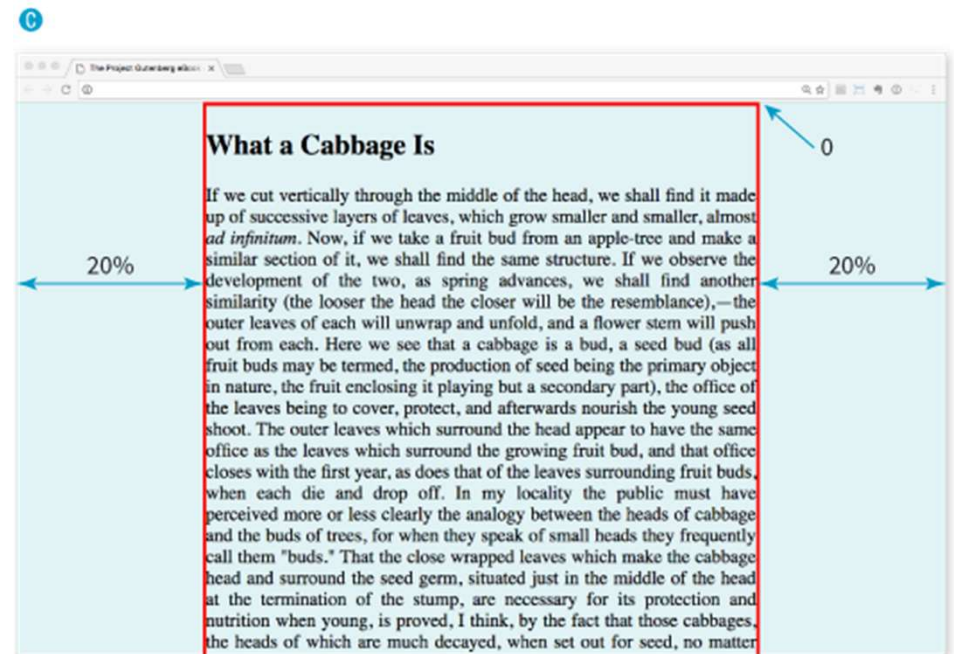
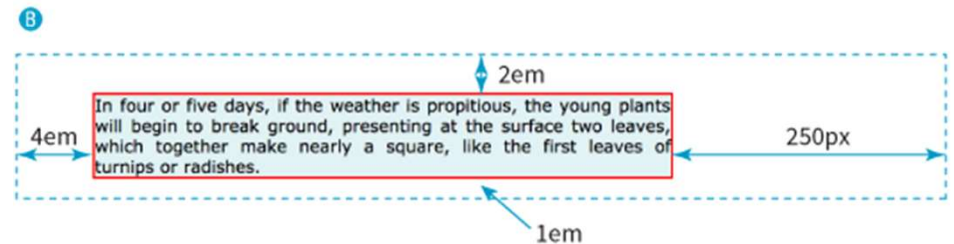
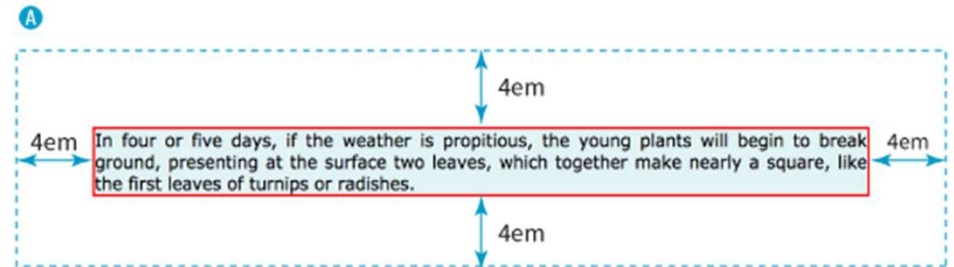
Values: *Length, percentage*

The **margin** is an amount of space added on the outside of the border. They keep elements from bumping into one another or the edge of the viewport.

The shorthand **margin** property works the same as the **padding** shorthand. Values are applied clockwise (TRouBLE order) and are repeated if fewer than 4 values are supplied.

Margins (cont'd)

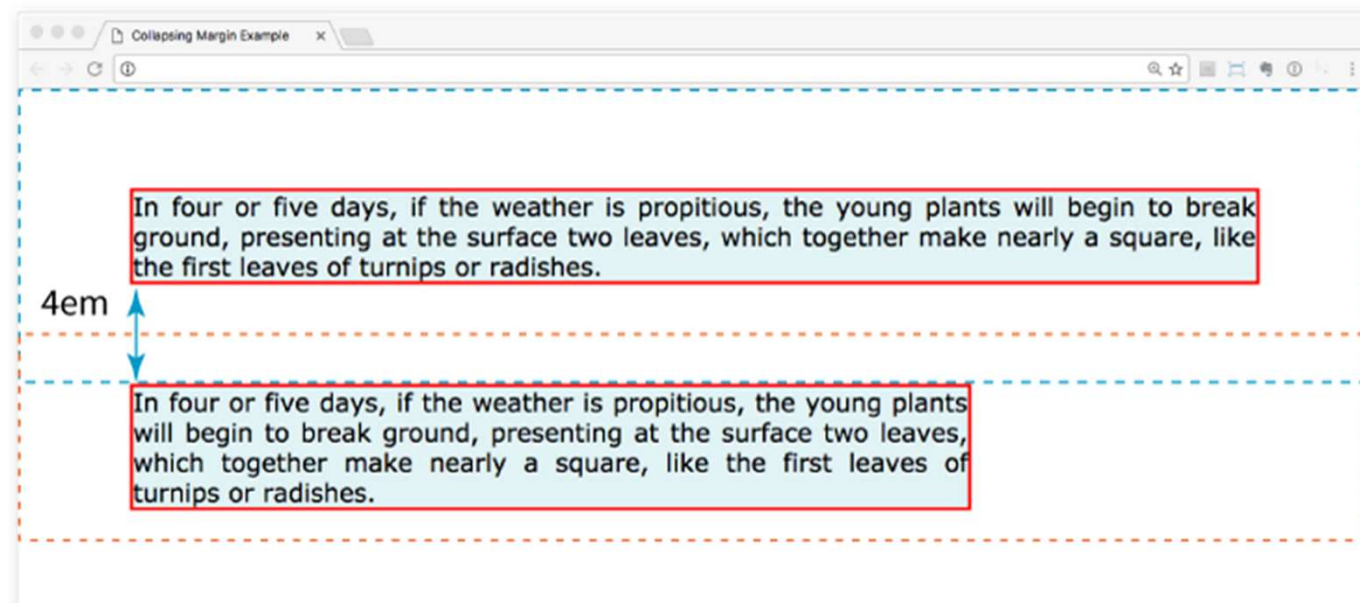
- A** `p#A {`
 `margin: 4em;`
 `border: 2px solid red;`
 `background: #e2f3f5;`
}
- B** `p#B {`
 `margin-top: 2em;`
 `margin-right: 250px;`
 `margin-bottom: 1em;`
 `margin-left: 4em;`
 `border: 2px solid red;`
 `background: #e2f3f5;`
}
- C** `body {`
 `margin: 0 20%;`
 `border: 3px solid red;`
 `background-color: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



Margins (cont'd)

Top and bottom margins of neighboring elements **collapse** (they overlap instead of accumulating).

The top element has a bottom margin of 4em. The bottom element has a top margin of 2em. The resulting margin is 4em (the largest value).



Assigning Display Types

display

Values: inline | block | run-in | flex | grid | flow | flow-root | list-item | table | table-row-group | table-header-group | table-footer-group | table-row | table-cell | table-column-group | table-column | table-caption | ruby | ruby-base | ruby-text | ruby-base-container | ruby-text-container | inline-block | inline-table | inline-flex | inline-grid | contents | none

Assigns a **display type** that determines how the element box behaves in layouts.

Examples:

- Make `li` (normally block elements) into inline elements so they line up in a horizontal menu: `nav li { display: inline; }`
- Make an anchor (`a`) element (normally inline) display as a block so you can give it a width and height: `nav li a { display: block; }`

Box Drop Shadows

`box-shadow`

Values: *horizontal-offset vertical-offset blur-distance spread-distance color inset, none*

Applies a drop shadow around the visible element box.

The values are a **horizontal** and **vertical offset**, optional **blur** and **spread** values (in pixels), a **color** value, and the option to make it appear **inset**.

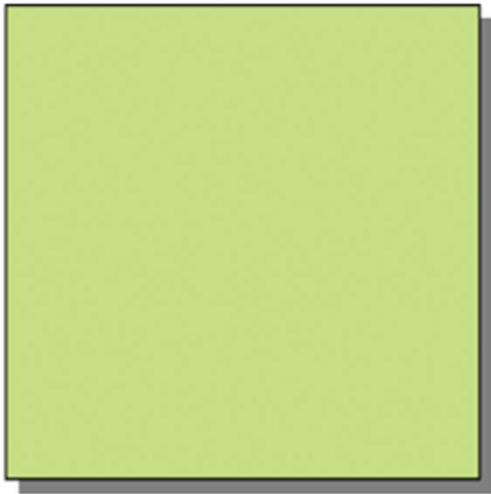
Box Drop Shadows (cont'd)

A `box-shadow: 6px 6px gray;`

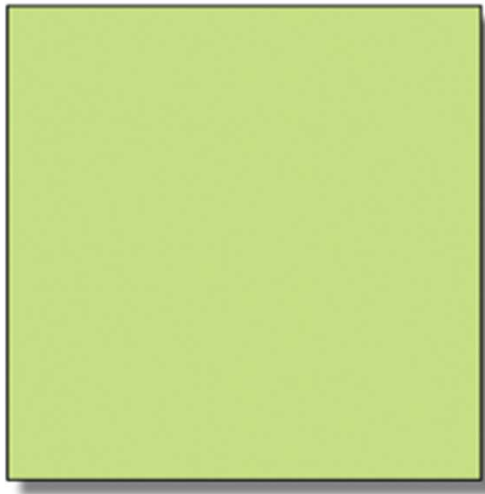
B `box-shadow: 6px 6px 5px gray; /* 5 pixel blur */`

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

A



B



C



Exercise

EXERCISE 14-2. Border tricks

In this exercise, we'll have some fun with borders on the Black Goose Bakery page. In addition to putting borders around content sections of the page, we'll use borders to beef up the headlines and as an alternative to underlines under links.

1. Open *bakery-styles.css* in a text editor if it isn't already. We'll start with the basics by using the shorthand **border** property to add a tan double rule around the **main** element. Add the new declaration to the existing rule for **main**:

```
main {  
  padding: 1em;  
  border: double 4px #EADD4;  
}
```

2. Now try out some **border-radius** properties to add generous rounded corners to the **main** and **aside** sections. A 25-pixel radius should do. Pixels are my choice over ems here because I don't want the radius to scale with the text. Start by adding this declaration to the styles for **main**:

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

And give just the top-right corner of the **aside** a matching rounded corner:

```
aside {  
  border-top-right-radius: 25px;  
}
```

EXERCISE 14-2. Continued

3. Just for fun (and practice), we'll add a decorative border on two sides of the baked goods headings (**h3**). Find the existing rule for **h3** elements and add a declaration that adds a 1-pixel solid rule on the top of the headline. Add another that adds a thicker 3-pixel solid rule on the left. I want the borders to be the same color as the text, so we don't need to specify the **border-color**. Finally, to prevent the text from bumping into the left border, add a little bit of padding (1em) to the left of the headline content:

```
h3 {  
  border-top: 1px solid;  
  border-left: 3px solid;  
  padding-left: 1em;  
}
```

4. The last thing we'll do is to replace the standard underline with a decorative bottom border under links. Start by turning off the underline for all links. Add this rule in the "link styles" section of the style sheet:

```
a {  
  text-decoration: none;  
}
```

Then add a 1-pixel dotted border to the bottom edge of links:

```
a {  
  text-decoration: none;  
  border-bottom: 1px dotted;  
}
```

As is often the case when you add a border to an element, it is a good idea to also add a little padding to keep things from bumping together:

```
a {  
  text-decoration: none;  
  border-bottom: 1px dotted;  
  padding-bottom: .2em;  
}
```

Now you can save the style sheet and reload *bakery.html* in the browser. **FIGURE 14-14** shows a detail of how your page should be looking so far.

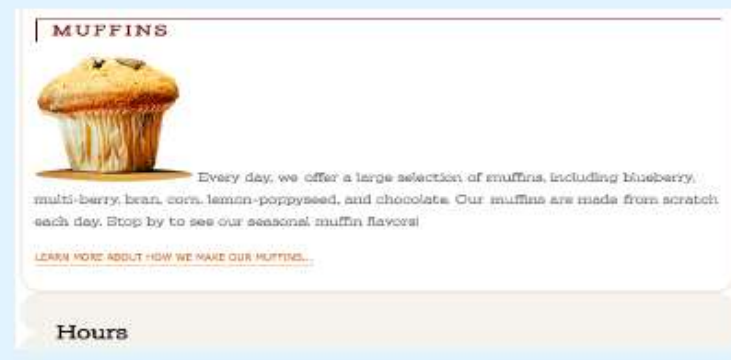


FIGURE 14-14. The results of our border additions.