CS 50 Web Design

APCSP Module 2: Internet

Unit 2: CSS (Chapter 11-13)

LECTURE 4: BASIC CSS STYLES

DR. ERIC CHOU

IEEE SENIOR MEMBER

Overview of CSS3

Chapter 11: CSS Hierarchy and Selectors
Chapter 12: Text, Image and Foreground (Contents)
Chapter 13: Color and Background (Contents)

Chapter 14: Box Model (Padding, Border, and Margin)
Chapter 15: Layout Management (Floating and Positioning)

Chapter 16: Layout Management (CSS Layout with Flexbox and Grid)
Chapter 17: Responsive Design (Page Level Planning)
Chapter 18: Transition, Transforms, and Animation
Chapter 19: CSS Techniques (Put Everything Together)

Objectives

- The benefits and power of CSS
- How HTML markup creates a document structure
- Writing style rules
- Attaching styles to the HTML document
- Big concepts: inheritance, specificity, the cascade, rule order, and the box model

Objectives

- Font properties
- Web fonts
- Advanced typography
- with CSS3
- Text line settings
- Text effects
- Selectors: descendent,
- ID, and class
- Specificity overview
- List styles

Objectives

- CSS color names
- RGB color values
- Foreground and background
- colors
- Tiling background images
- Color gradients
- Pseudo-class, pseudo-element,
- and attribute selectors
- External style sheets

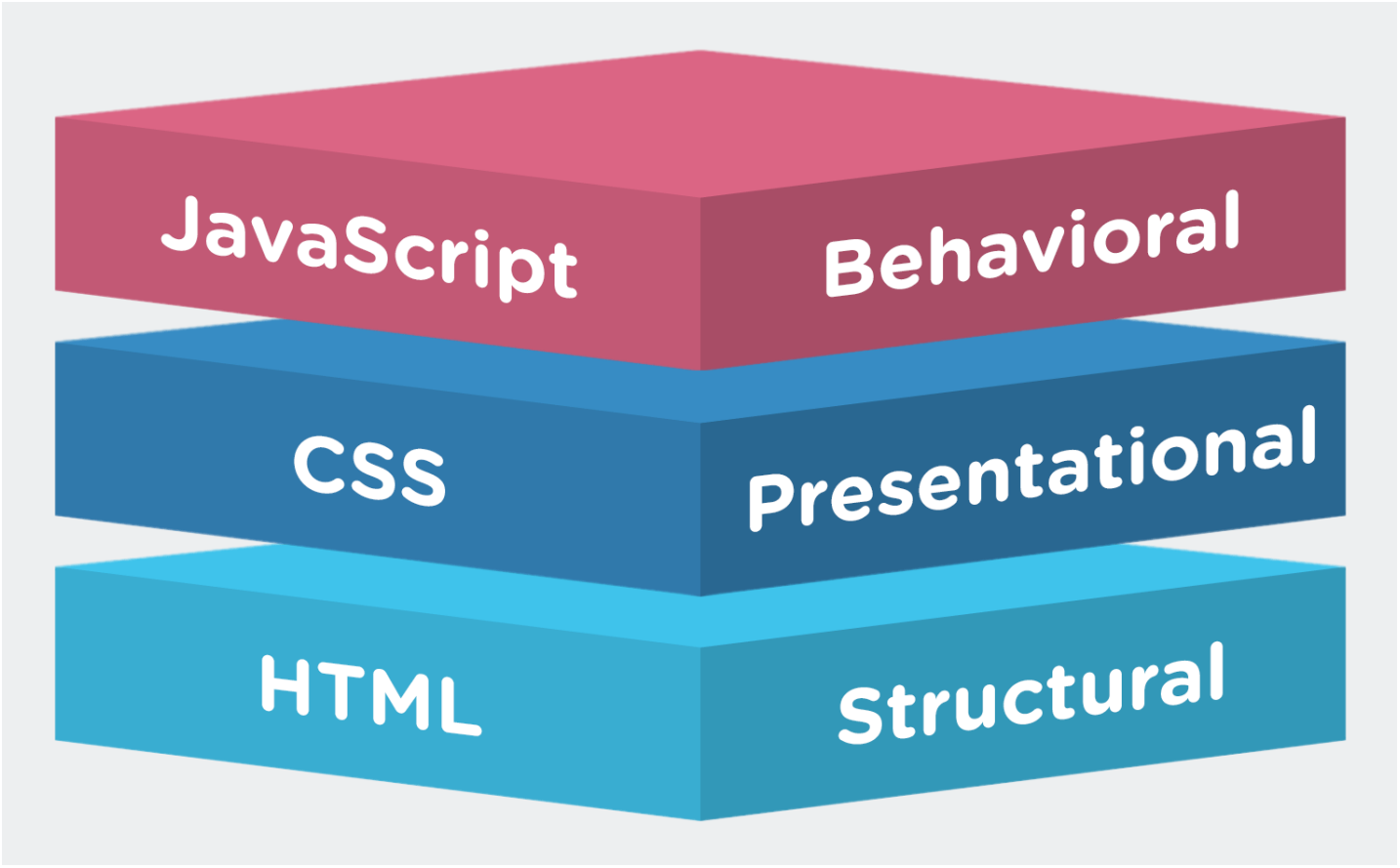
Overview

SECTION 1

What is CSS3?

- Cascading Style Sheets (CSS) is the W3C standard for defining the presentation of documents written in HTML, and in fact, any **XML** language .
- Presentation, again, refers to the way the document is displayed or delivered to the user, whether on a computer screen, a cell phone display, printed on paper, or read aloud by a screen reader.
- With style sheets handling the presentation, HTML can handle the business of defining document structure and meaning, as mentioned.
- Public resource that can use:
  - http://www.dynamicdrive.com/style/
  - http://www.free-css-templates.com/

Marking up the document



The Benefits of CSS

- Precise type and layout controls
- Less work
- More Accessible sites
- Reliable browser support

How Style Sheets Work

- Marking up the document
- Writing the rules
- Attaching styles to the document

How Style Sheets Work?

1. Marking Up the Document

- You know that it is important to choose elements that accurately describe the meaning of the content. You also heard me say that the markup creates the structure of the document, sometimes called the **structural layer**, upon which the **presentation layer** can be applied.
- In this and the upcoming chapters, you'll see that having an understanding of your document's structure and the relationships between elements is central to your work as a style sheet author.
- In the exercises throughout this chapter, you will get a feel for how simple it is to change the look of a document with style sheets.

How Style Sheets Work?

2. Writing the Rules

- A style sheet is made up of one or more style instructions (called style rules) that describe how an element or group of elements should be displayed.
- The first step in learning CSS is to get familiar with the parts of a rule. As you'll see, they're fairly intuitive to follow. Each rule selects an element and declares how it should look.

How Style Sheets Work?

2. Writing the Rules

- In CSS terminology, the two main sections of a rule are the selector that identifies the element or elements to be affected, and the declaration that provides the rendering instructions.
- The declaration, in turn, is made up of a property (such as **color**) and its value (**green**), separated by a colon and a space.

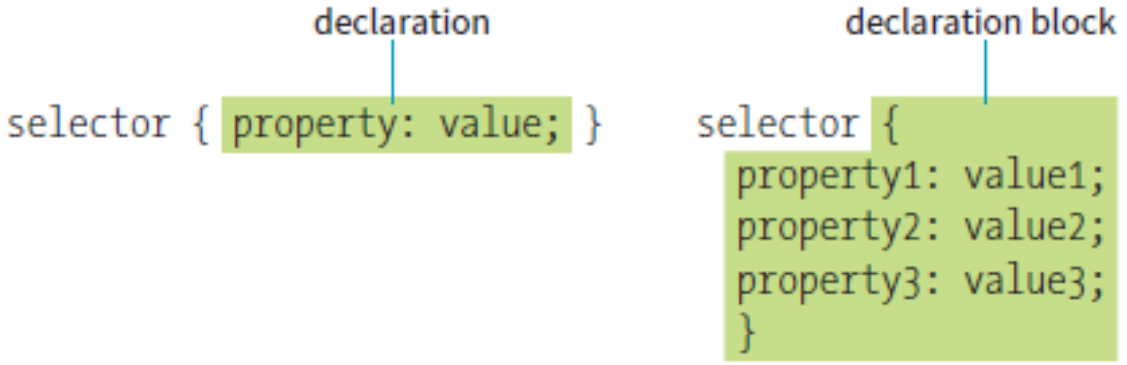


FIGURE 11-3. The parts of a style rule.

Selectors

- In the previous small style sheet example, the **h1** and **p** elements are used as selectors. This is called an element type selector, and it is the most basic type of selector. The properties defined for each rule will apply to every **h1** and **p** element in the document, respectively.
- Another type of selector is an ID selector, which selects an element based on the value of an element's **id** attribute. It is indicated with the **#** symbol. For example, the selector **#recipe** targets an element with **id="recipe"**.

Selectors

- In upcoming chapters, I'll introduce you to more sophisticated selectors that you can use to target elements, including ways to select groups of elements, and elements that appear in a particular context. See the "Selectors in this Book" sidebar for details.
- Mastering selectors—that is, choosing the best type of selector and using it strategically—is an important step in mastering CSS.

Declarations

- The declaration is made up of a **property/value** pair. There can be more than one declaration in a single rule; for example, the rule for the p element shown earlier in the code example has both the **font-size** and **font-family** properties.
- Each declaration must end with a semicolon to keep it separate from the following declaration.

```
p {
   font-size: large;
   font-family: sans-serif;
}
```

Declarations

Properties

- The heart of style sheets lies in the collection of standard properties that can be applied to selected elements. The complete CSS specification defines dozens of **properties** for everything from text indents to how table headers should be read aloud. This book covers the most common and best-supported properties that you can begin using right away.

Declarations

Values

- **Values** are dependent on the property. Some properties take length measurements, some take color values, and others have a predefined list of keywords.
- When you use a property, it is important to know which values it accepts; however, in many cases, simple common sense will serve you well.
- Authoring tools such as Dreamweaver or Visual Studio provide hints of suitable values to choose from. Before we move on, why not get a little practice writing style rules yourself in EXERCISE 11-2?

How Style Sheets Work?

3. Attaching the Styles to the Document

- External style sheets

  An external style sheet is a separate, text-only document that contains a number of style rules. It must be named with the **.css** suffix. The **.css** document is then **linked** to (via the link element) or imported (via an **@import** rule in a style sheet) into one or more HTML documents.

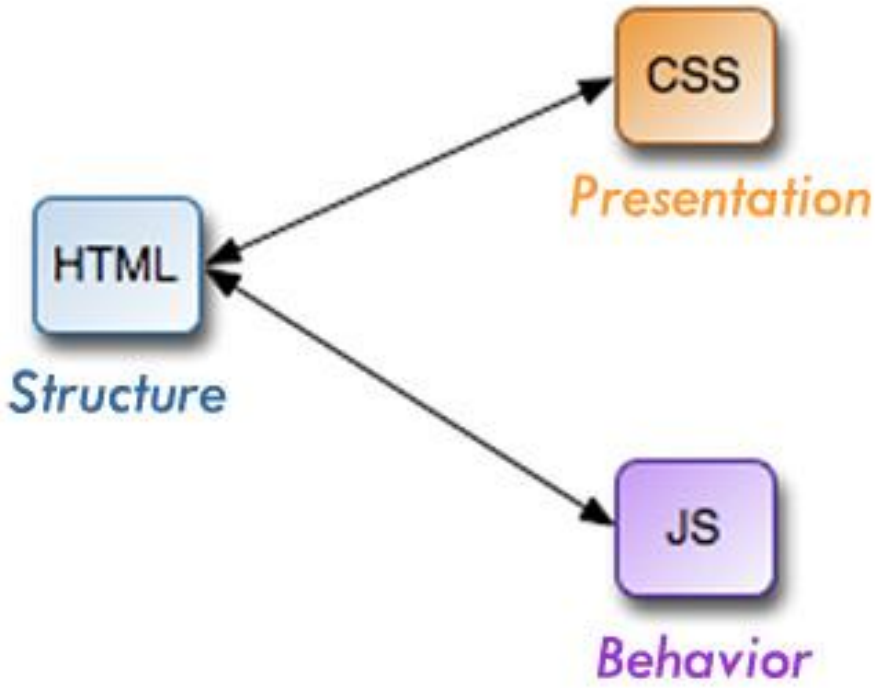
- Embedded style sheets

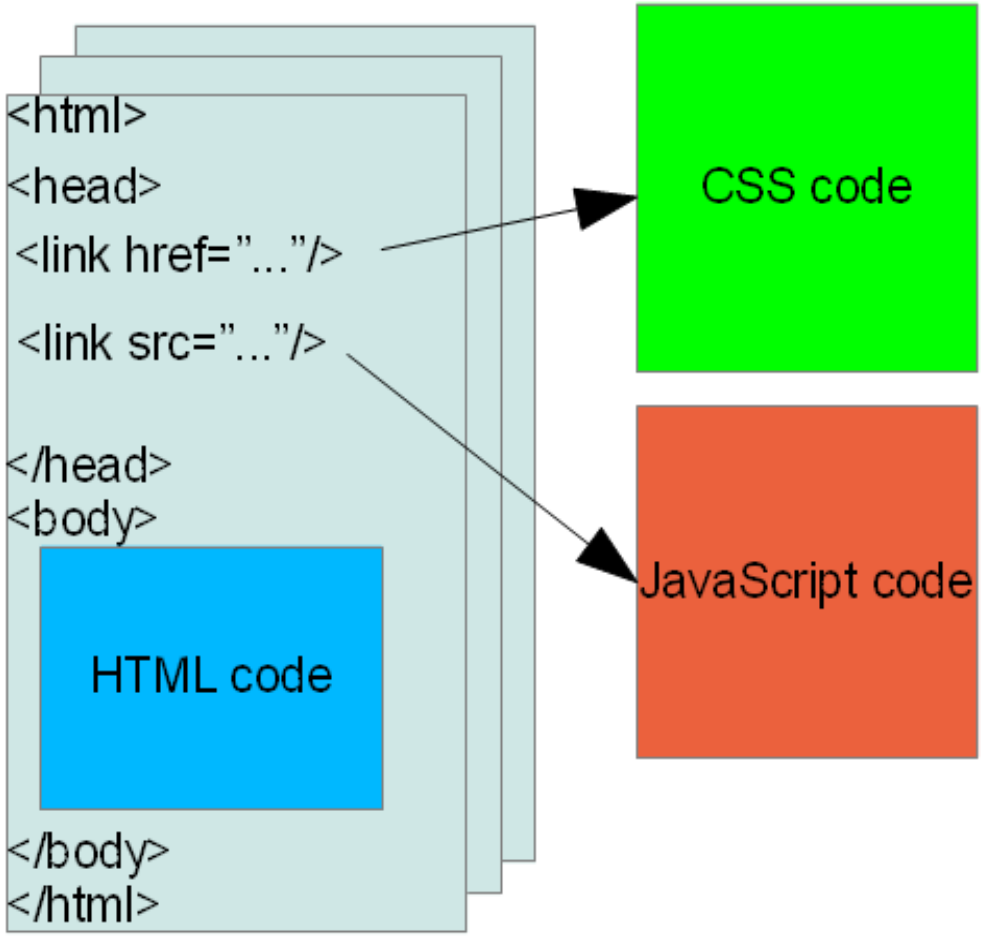
```
<head>
  <title>Required document title here</title>
  <style>
    /* style rules go here */
  </style>
</head>
```

- Inline styles

```
<h1 style="color: red">Introduction</h1>
```

File Structure





Object Hierarchy

SECTION 2

Inheritance by Ownership

- Are your eyes the same color as your parents'? Did you inherit their hair color? Well, just as parents pass down traits to their children, styled HTML elements pass down certain style properties to the elements they contain.
- Notice in EXERCISE 11-1, when we styled the **p** elements in a large, sans-serif font, the em element in the second paragraph became large and sans-serif as well, even though we didn't write a rule for it specifically (FIGURE 11-5). That is because the **em** element inherited the styles from the paragraph it is in.
- Inheritance provides a mechanism for styling elements that don't have any explicit styles rules of their own.

Unstyled paragraph

I've been doing a lot of *talking* about cooking

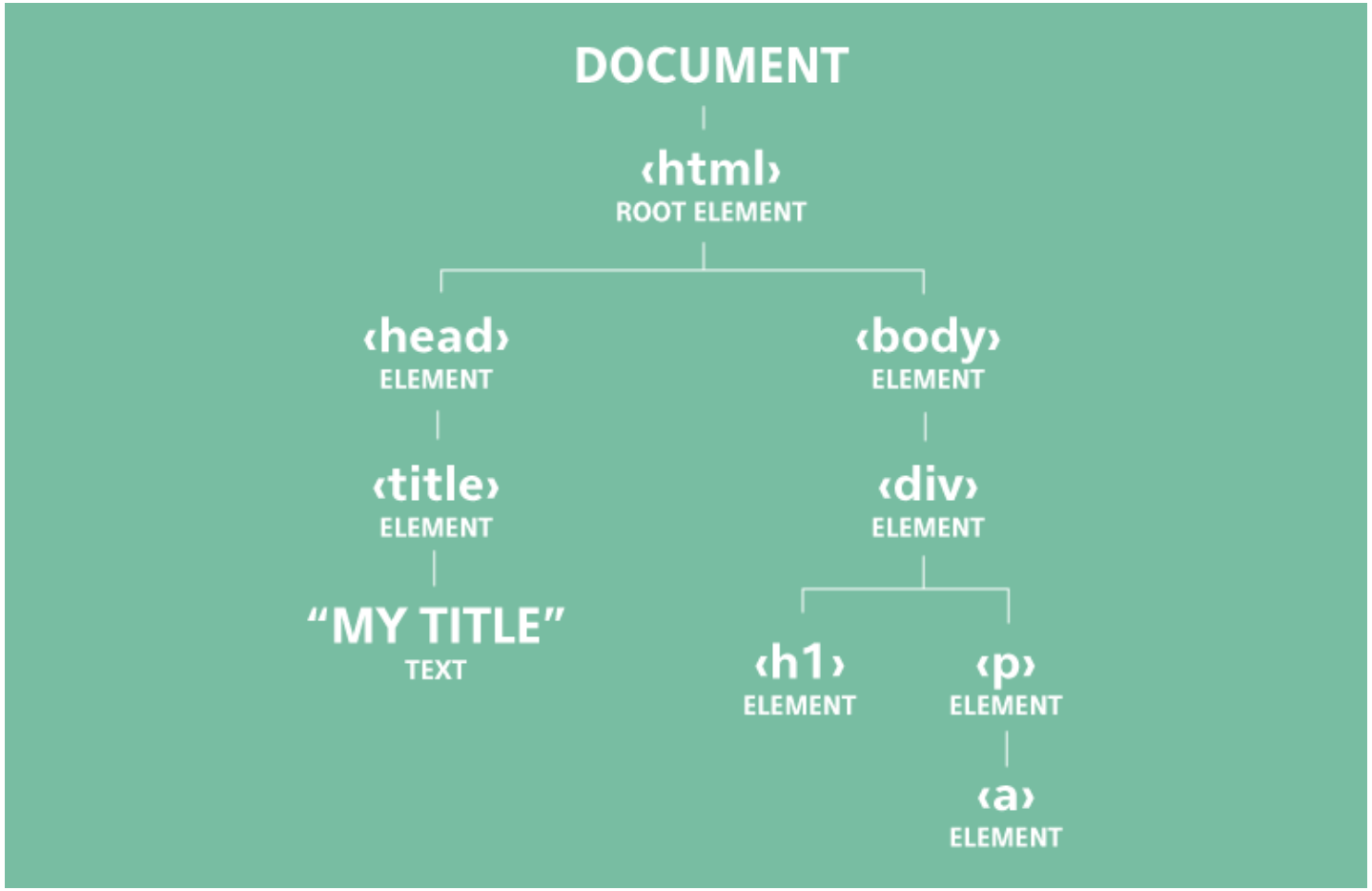
Paragraph with styles applied

I've been doing a lot of *talking* about cooking

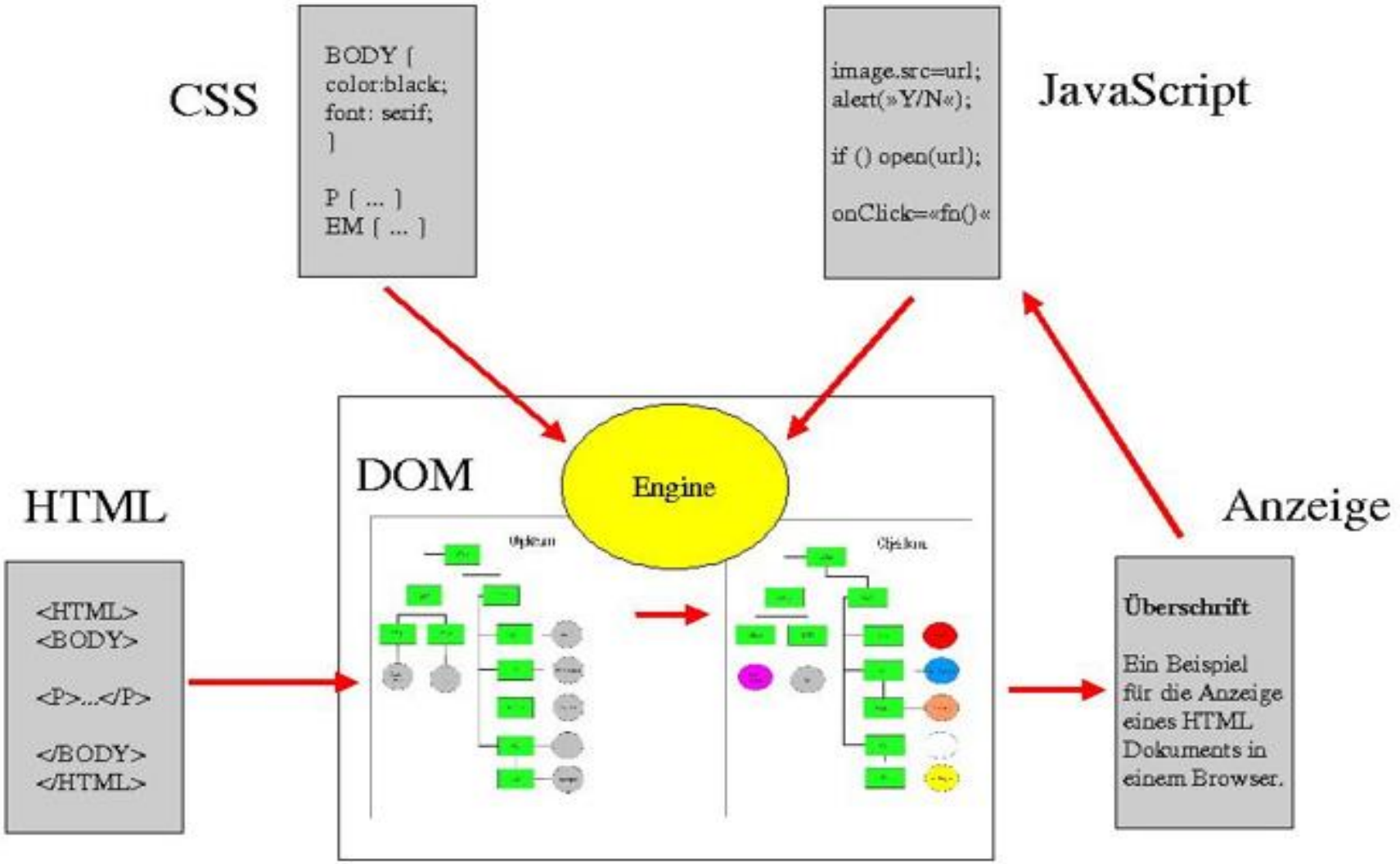
The `em` element is large and sans-serif even though it has no style rules of its own. It *inherits* styles from the paragraph that contains it.

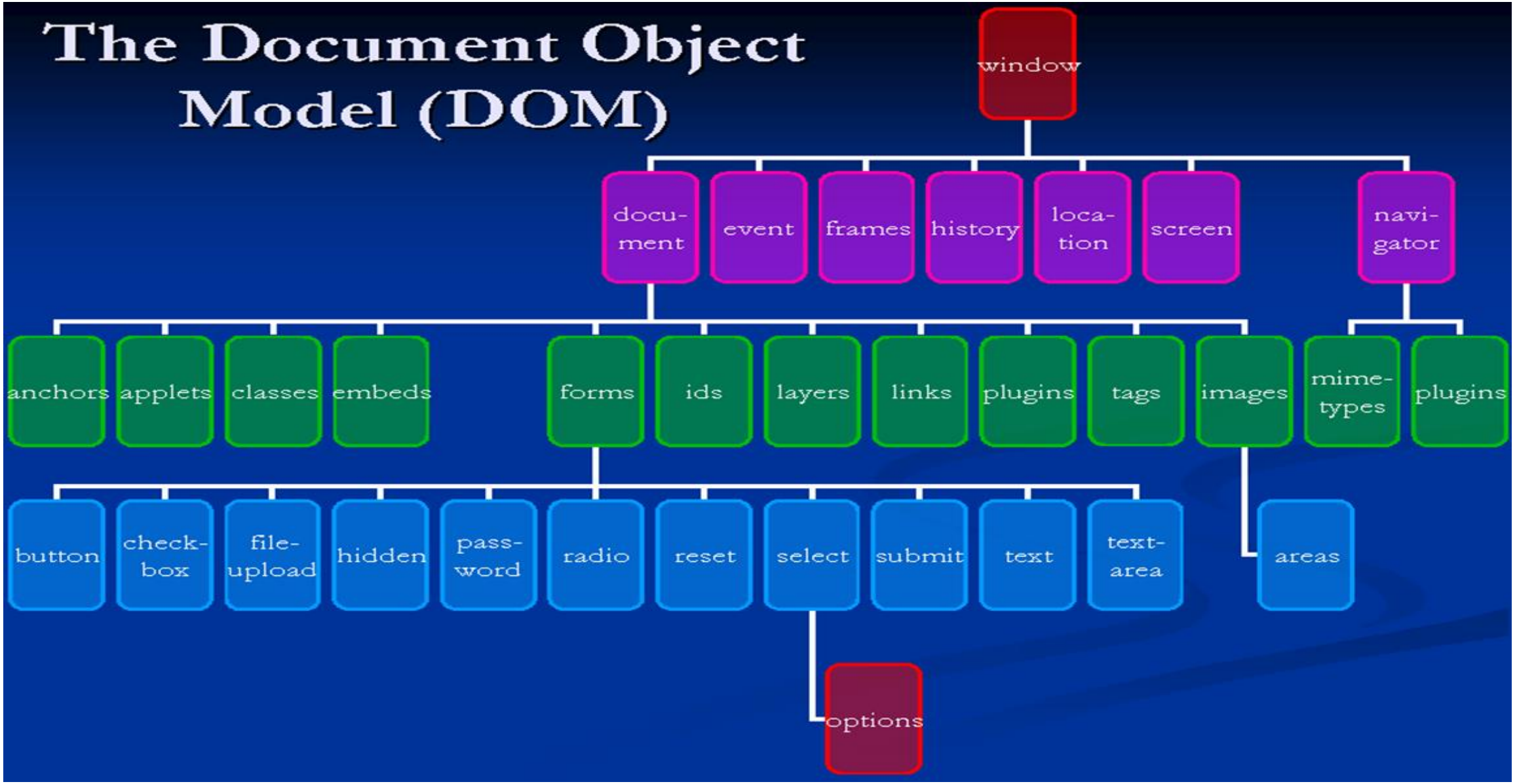
FIGURE 11-5. The `em` element inherits styles that were applied to the paragraph.

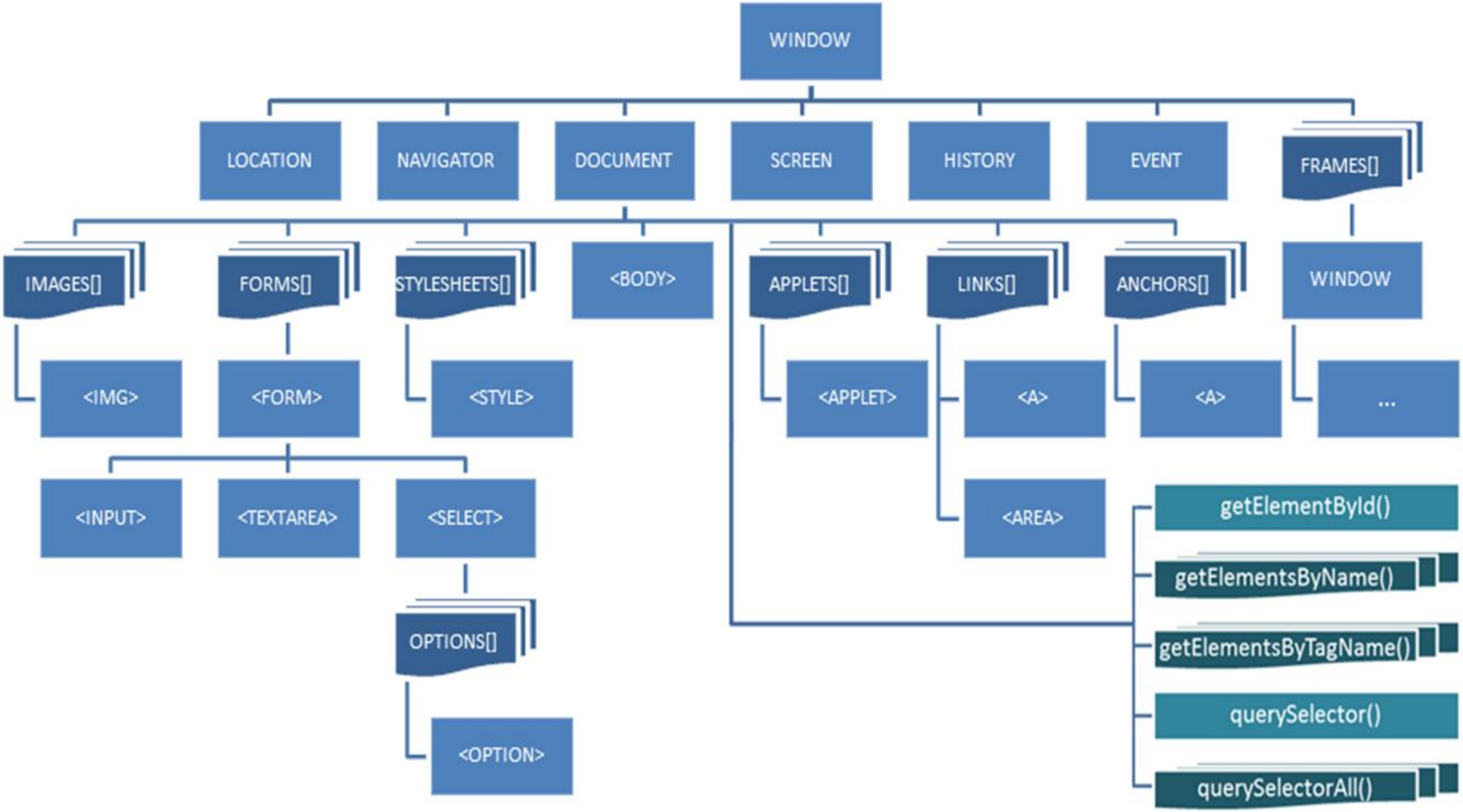
Structural View by HTML



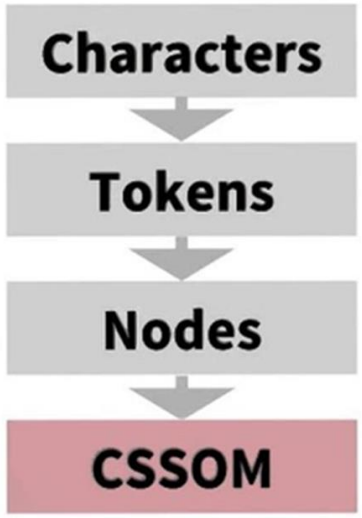
HTML, CSS, JavaScript und DOM

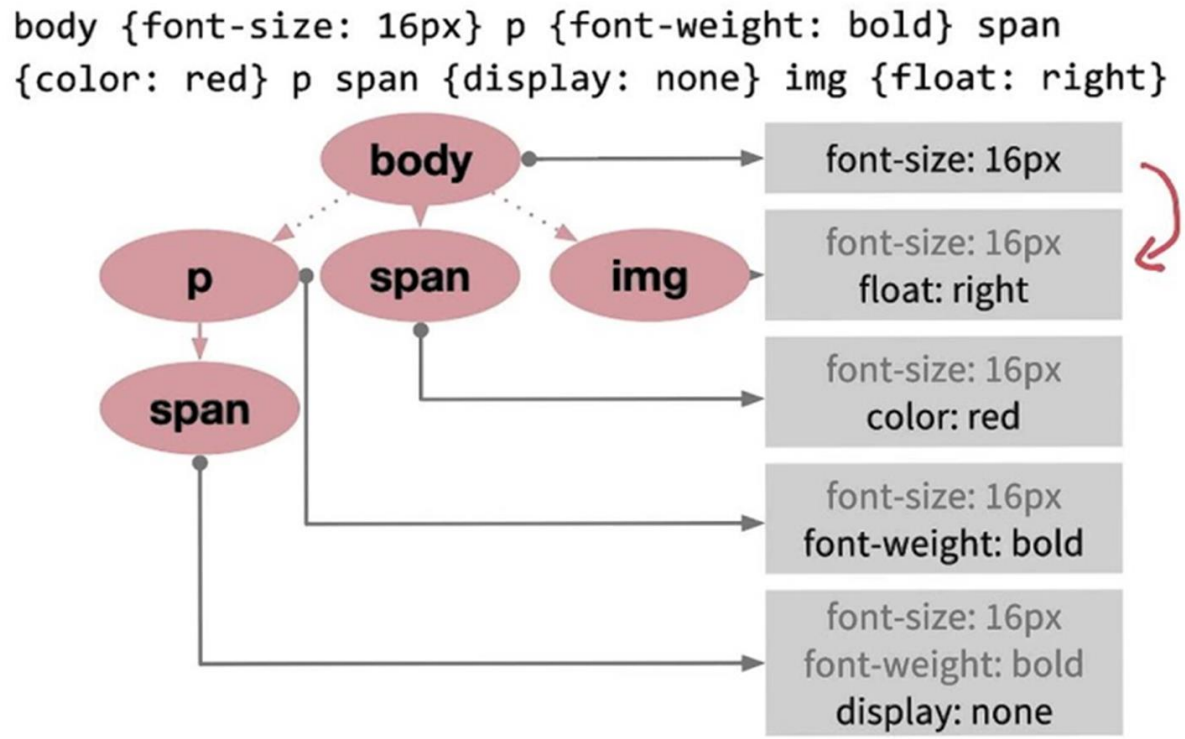



The Document Object Model (DOM)
window
docu-ment
event
frames
history
loca-tion
screen
navi-gator
anchors
applets
classes
embeds
forms
ids
layers
links
plugins
tags
images
mime-types
plugins
button
check-box
file-upload
hidden
pass-word
radio
reset
select
submit
text
text-area
areas
options


WINDOW
LOCATION
NAVIGATOR
DOCUMENT
SCREEN
HISTORY
EVENT
FRAMES[]
IMAGES[]
FORMS[]
STYLESHEETS[]
<BODY>
APPLETS[]
LINKS[]
ANCHORS[]
WINDOW
<IMG>
<FORM>
<STYLE>
<APPLET>
<A>
<A>
...
<INPUT>
<TEXTAREA>
<SELECT>
<AREA>
OPTIONS[]
<OPTION>
getElementById()
getElementsByName()
getElementsByTagName()
querySelector()
querySelectorAll()

Demo Project CSS-Hierarchy





Document Structure

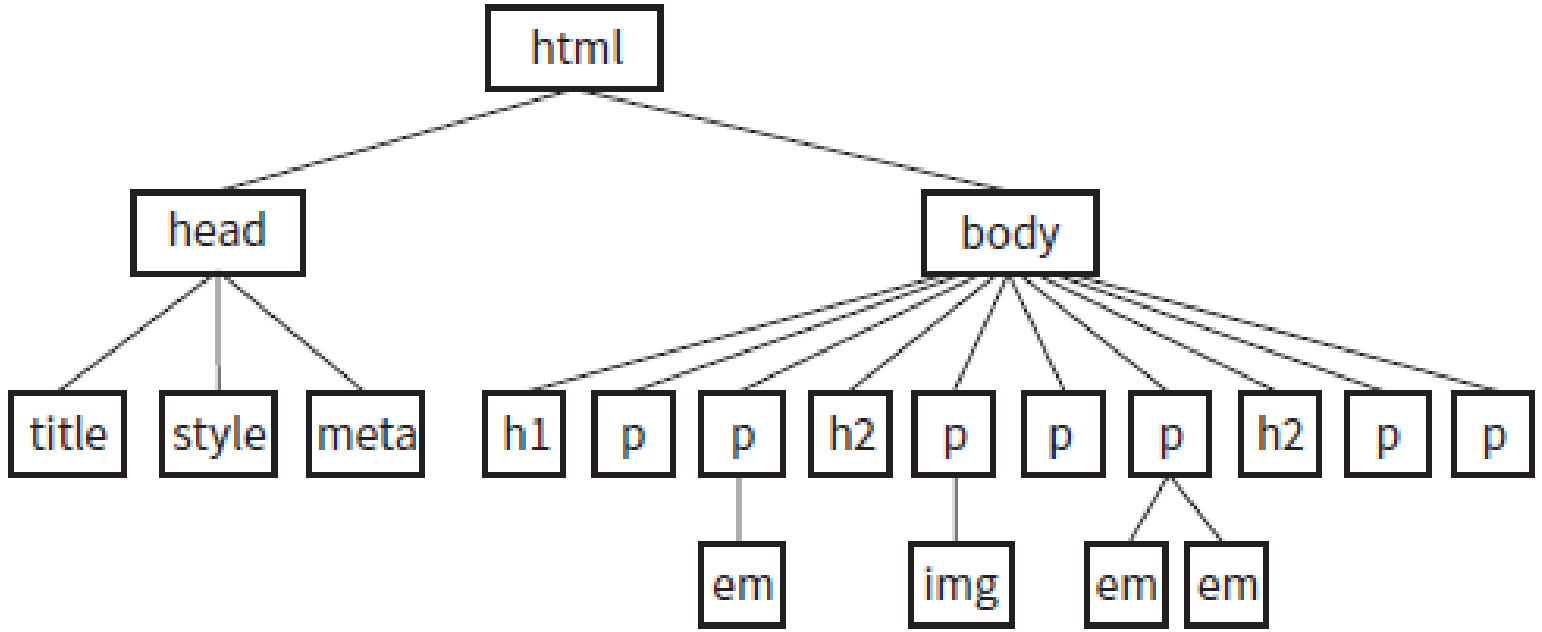


FIGURE 11-6. The document tree structure of the sample document, *cooking.html*.

Document Structure

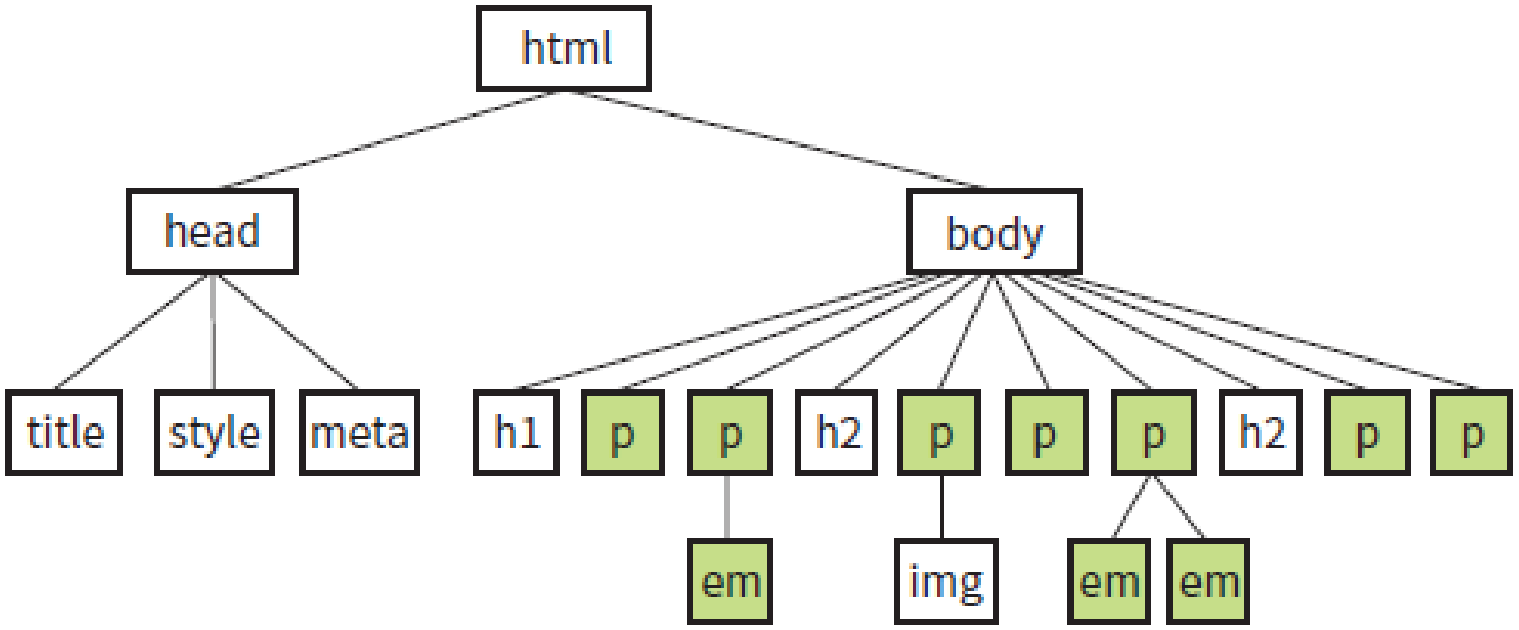


FIGURE 11-7. Certain properties applied to the **p** element are inherited by their children.

Document Structure

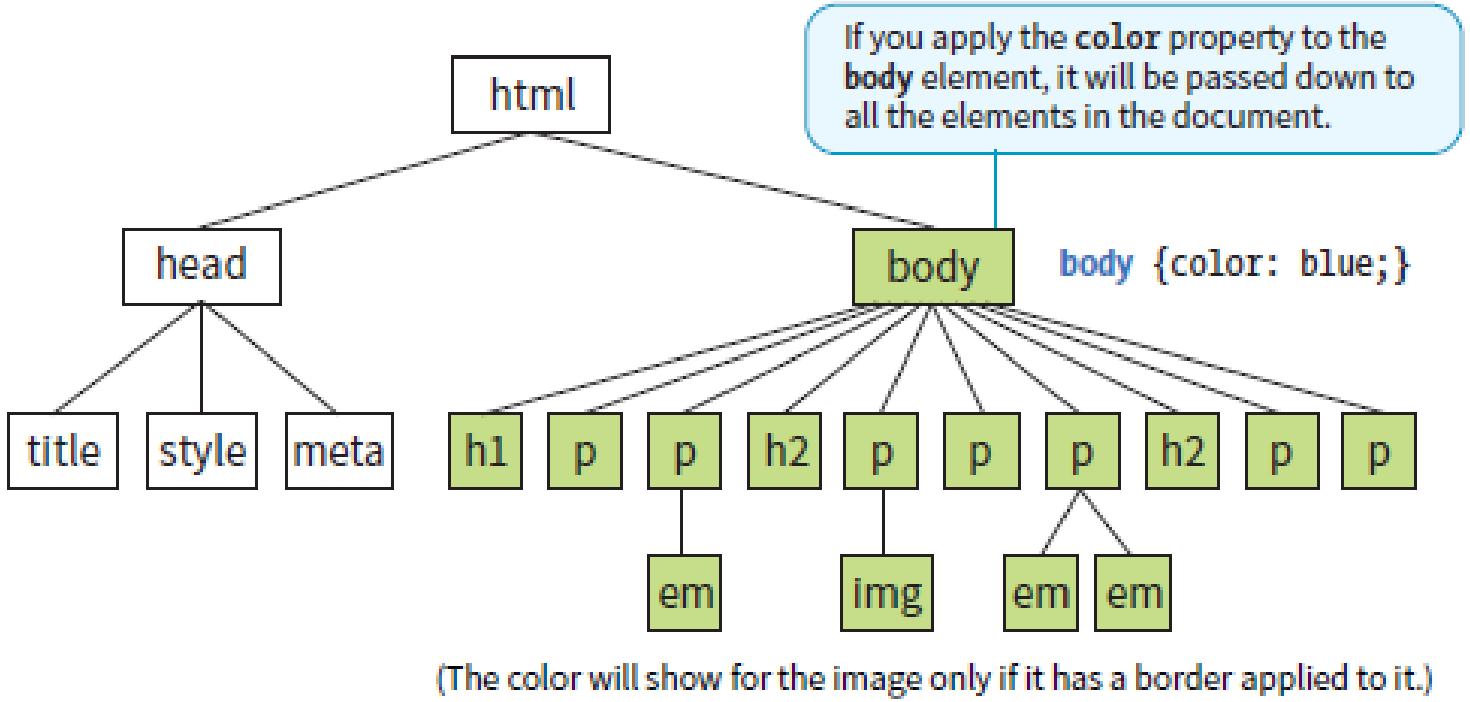
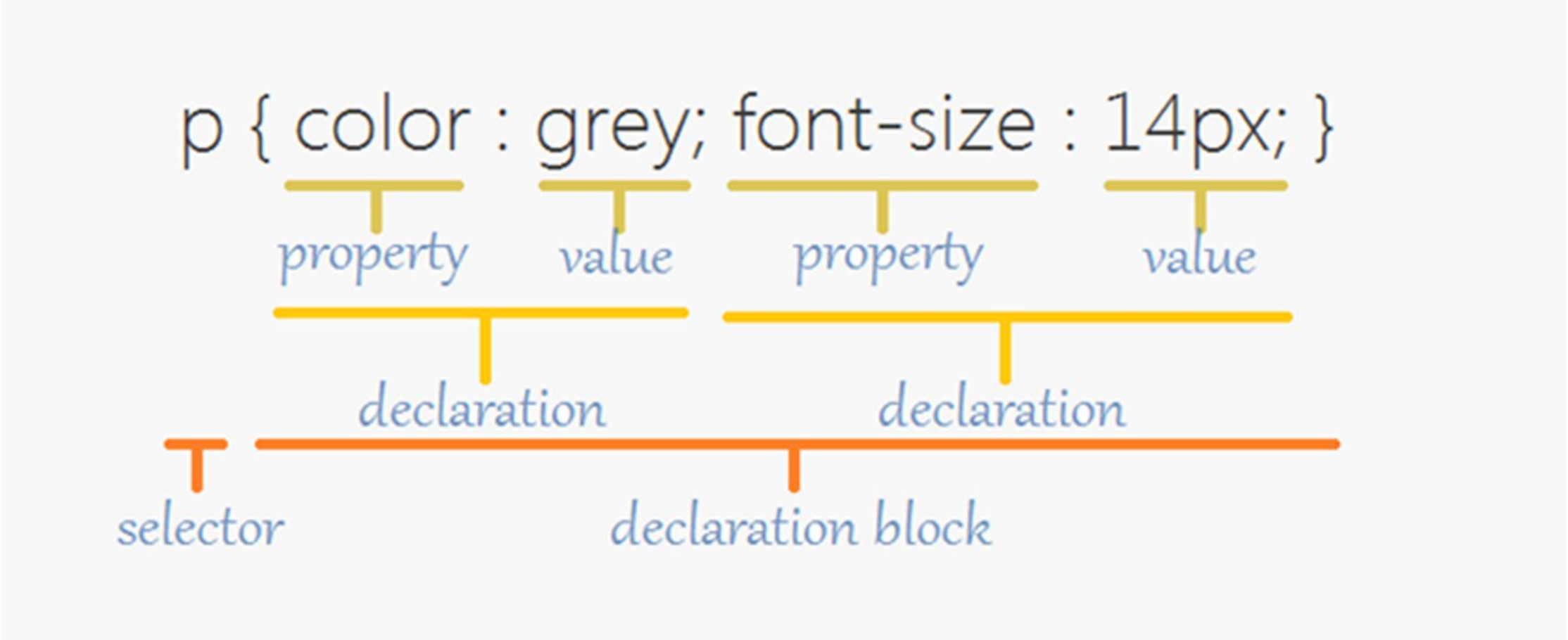


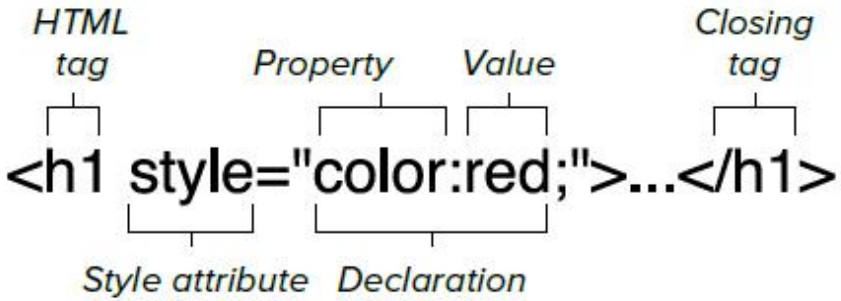
FIGURE 11-8. All the elements in the document inherit certain properties applied to the **body** element.

Writing CSS

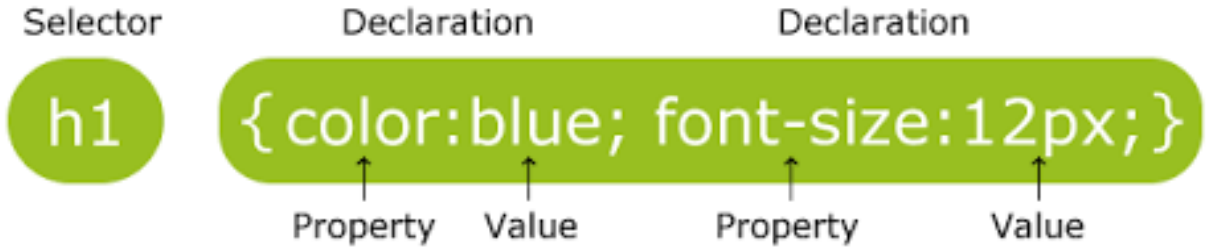
SECTION 3

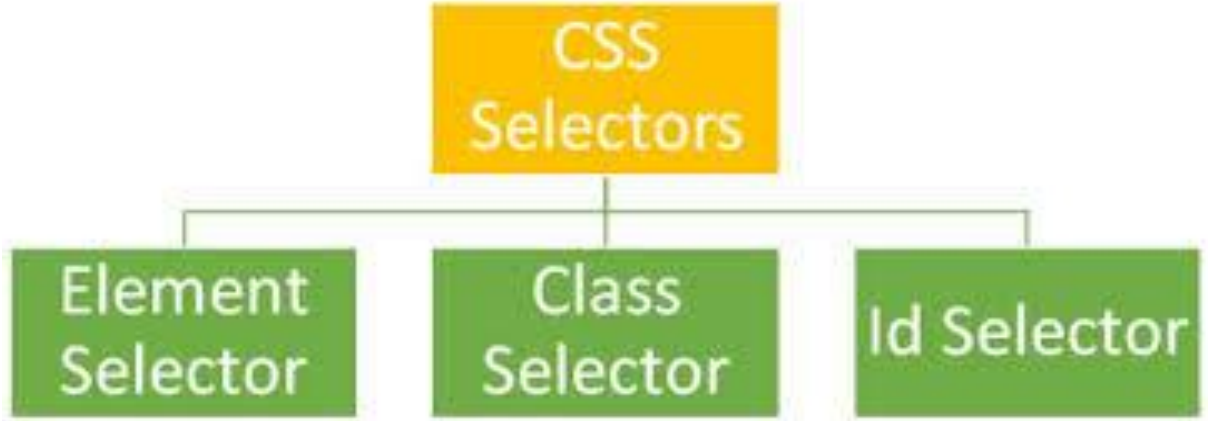

p { color : grey; font-size : 14px; }
property
value
property
value
declaration
declaration
selector
declaration block

CSS Selectors



The general syntax for defining styles directly in an HTML tag.





CSS Selectors

- **Element selector:**

The element selector selects elements based on the element name.

- **Example:**

```
<p > This is a paragraph </p>

p {
    text-align:center;
    color:red;
}
```

You can select all <p> elements on a page like this: (all <p> elements will be center-aligned, with a red text color)

- **Class selector:**

The class selector finds elements with the specific class.

- **Example:**

```
.center {
  text-align: center;
  color: red;
}
```

You can specify that only specific HTML elements should be affected by a class.

- **Id selector:**

The id selector uses the id attribute of an HTML tag to find the specific element.

- **Example:**

```
#para1 {
  text-align: center;
  color: red;
}
```

The style rule below will be applied to the HTML element with id="para1":

Writing CSS Style

- **Selectors**: the selector shown above is called Element Selector which means the style definition is for all <p> tags.
- **Declaration**: each property:value pair is called a declaration. A style definition could include several declarations which is named as declaration block.
- **Elements in a Element**:

  ul li { ... Style Definition ...}
- **Elements in a Element with id**:

  #unorderedlistitem li {... Style Definition ....}

Writing CSS Style

- **Class (group of Elements) in an Element:**

  form.age {… Style Definition …}

- **CSS Selector Structure:**

  grandparent parent me son grandchild {… Style Definition …}

- Work on Ex. 11-1

Attaching the Styles to the Documents

External Style Sheets:

<LINK **REL=“stylesheet”** HREF="style.css" TYPE="text/css" MEDIA=screen>

Embedded Style Sheets:

```
<STYLE TYPE="text/css" MEDIA=screen>
    <!-- BODY  { background: url(foo.gif) red; color: black }
         P EM  { background: yellow; color: black }
         .note { margin-left: 5em; margin-right: 5em } -->
</STYLE>
```

Inline Style Definition:

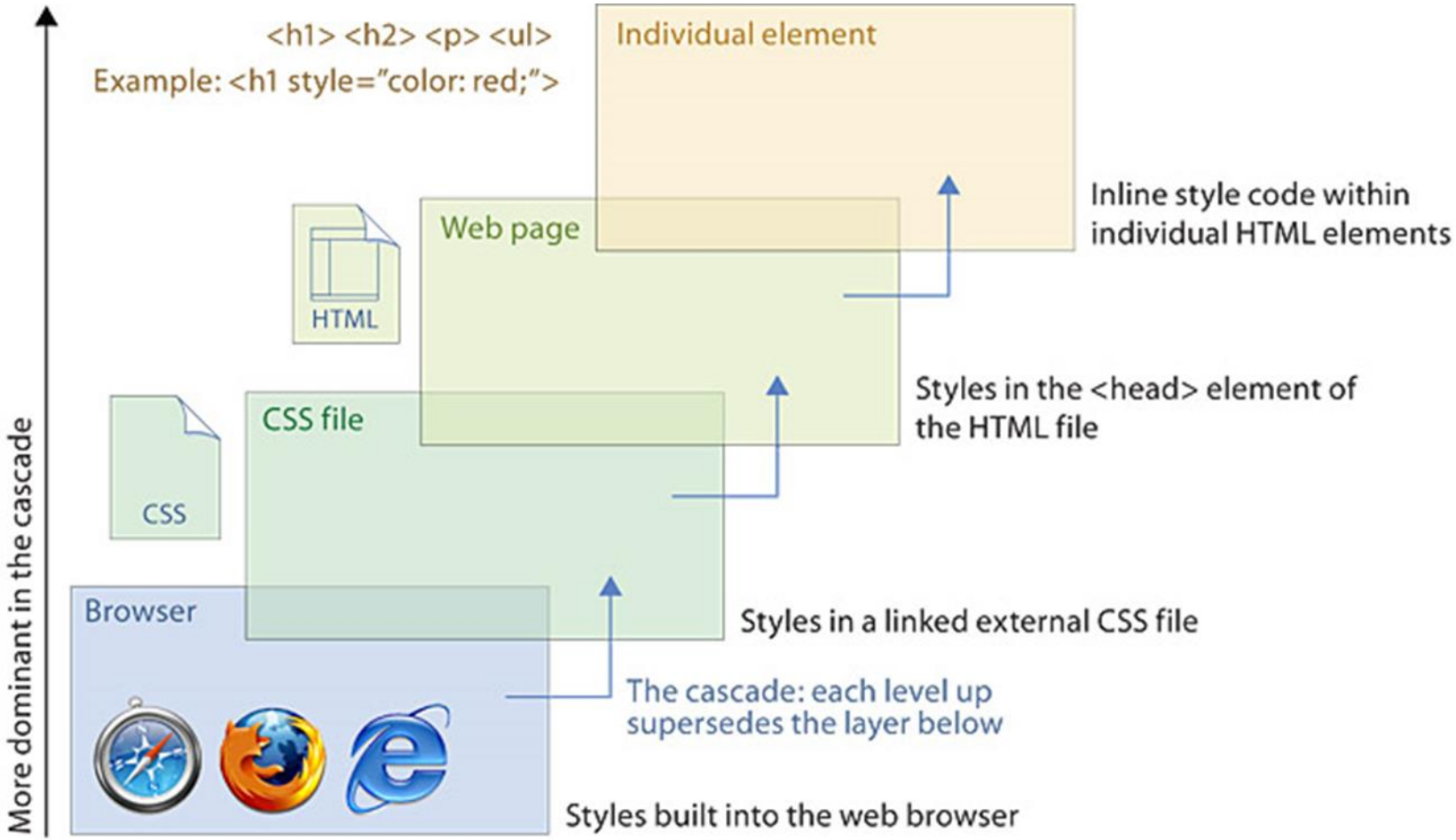
```
<P STYLE="color: red; font-family: 'New Century Schoolbook', serif">
This paragraph is styled in red with the New Century Schoolbook font, if available.</P>
```

Attaching the Styles to the Documents

Imported Style Sheets:

```
<style type="text/css" media="screen, projection">
<!--  @import url(http://www.htmlhelp.com/style.css);
      @import url(/stylesheets/punk.css);
      DT { background: yellow; color: black }-->
</style>
```

Rule Order

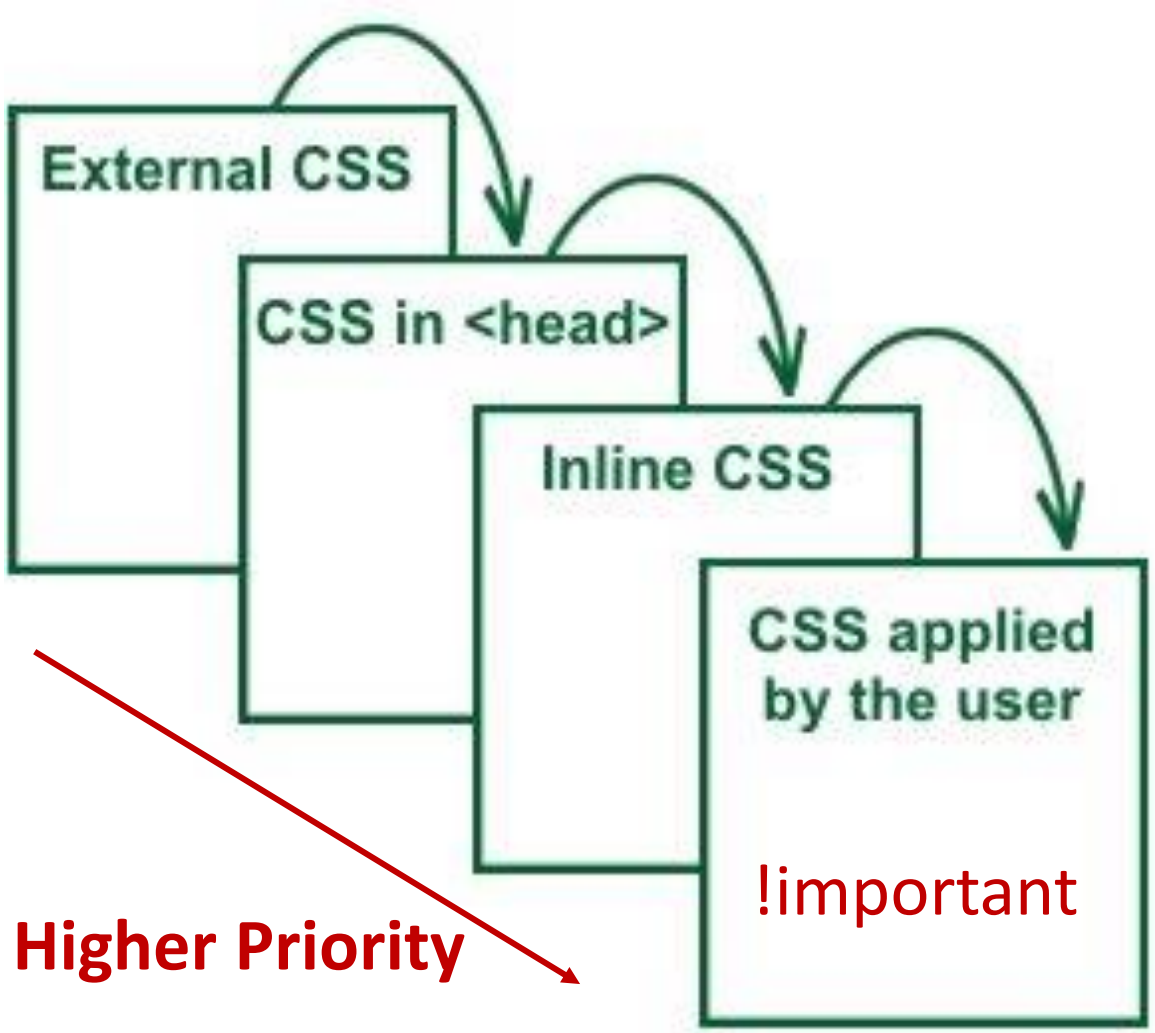

<h1> <h2> <p>
Example: <h1 style="color: red;">
Individual element
Inline style code within individual HTML elements
Web page
HTML
Styles in the <head> element of the HTML file
CSS file
CSS
Browser
Styles in a linked external CSS file
The cascade: each level up supersedes the layer below
Styles built into the web browser
More dominant in the cascade

Style Rule Hierarchy

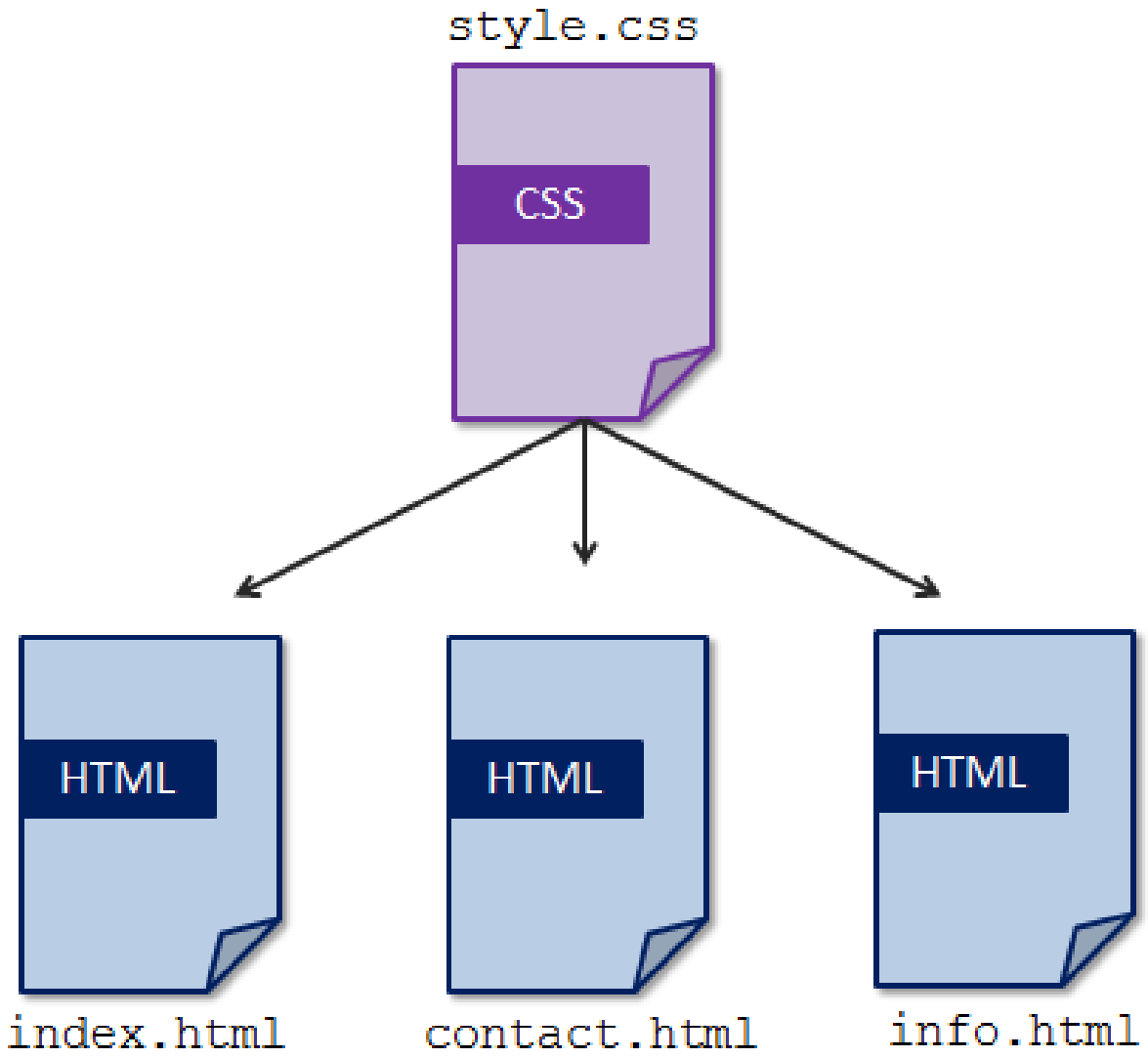
Style information can come from various origins, listed here from highest priority to lowest. In other words, items higher in the list override items below.

- Any style rule marked **!important** by the reader (user)
- Any style rule marked **!important** by the author
- Style sheets written by the author
- Style sheets created by the reader (user)
- Browser's default style rules ("user agent style sheet")

Style Rule Hierarchy



Cascading (2nd Meaning) One Sheet and for All

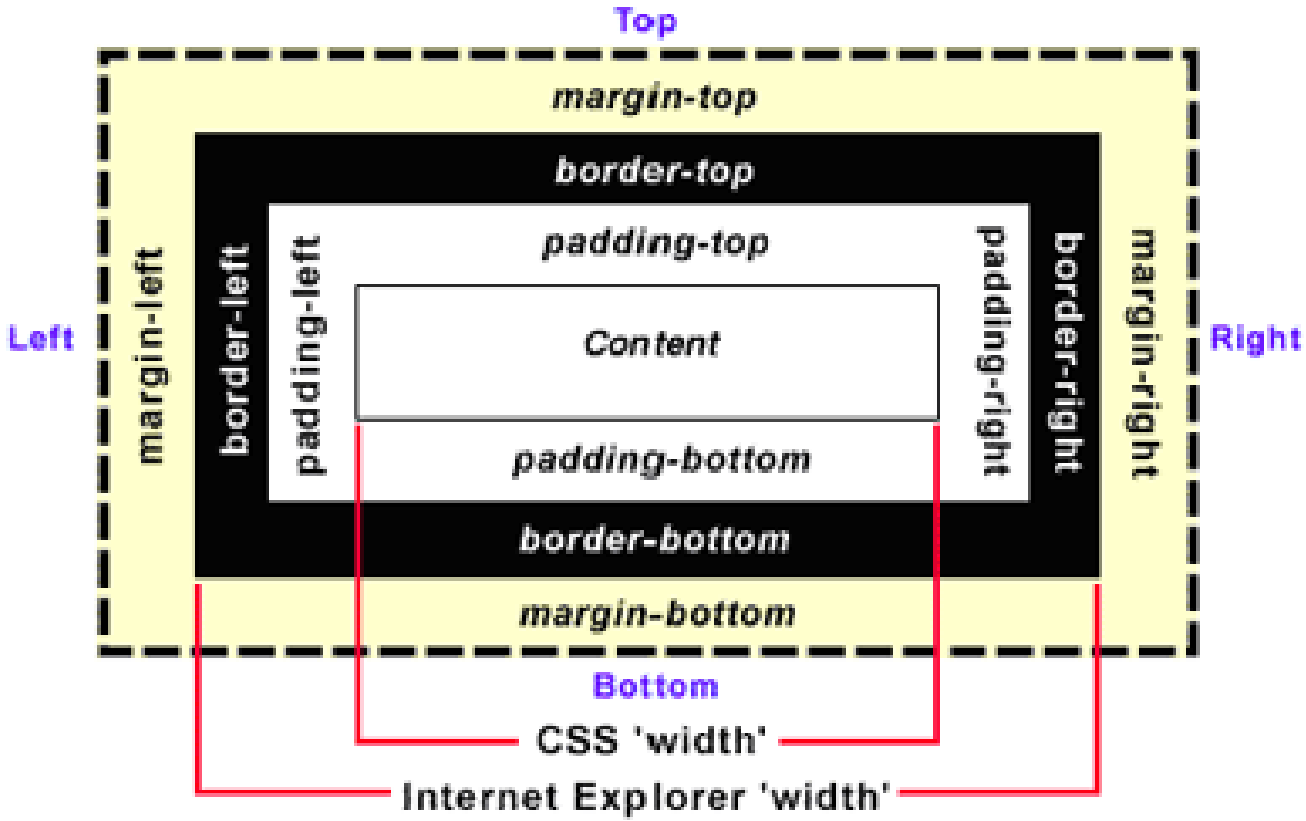


Inheritance and Cascading Priority

- The cascade refers to what happens when several sources of style information view for control of the elements on a page: style information is passed down **(“cascades” down)** until it is overridden by a style command with more weight. More weight means higher priority or lower in the element tree.
- !important is a indicator to prevent a specific rule from being overridden.

Box Model

SECTION 4



CSS Box model

Grouped selectors (all selectors will be affected)

h1,h2,p,em,img { border: 1px solid blue;}

```
h1 { border: 1px solid blue; }
h2 { border: 1px solid blue; }
p { border: 1px solid blue; }
em { border: 1px solid blue; }
img { border: 1px solid blue; }
```

Rules around all elements

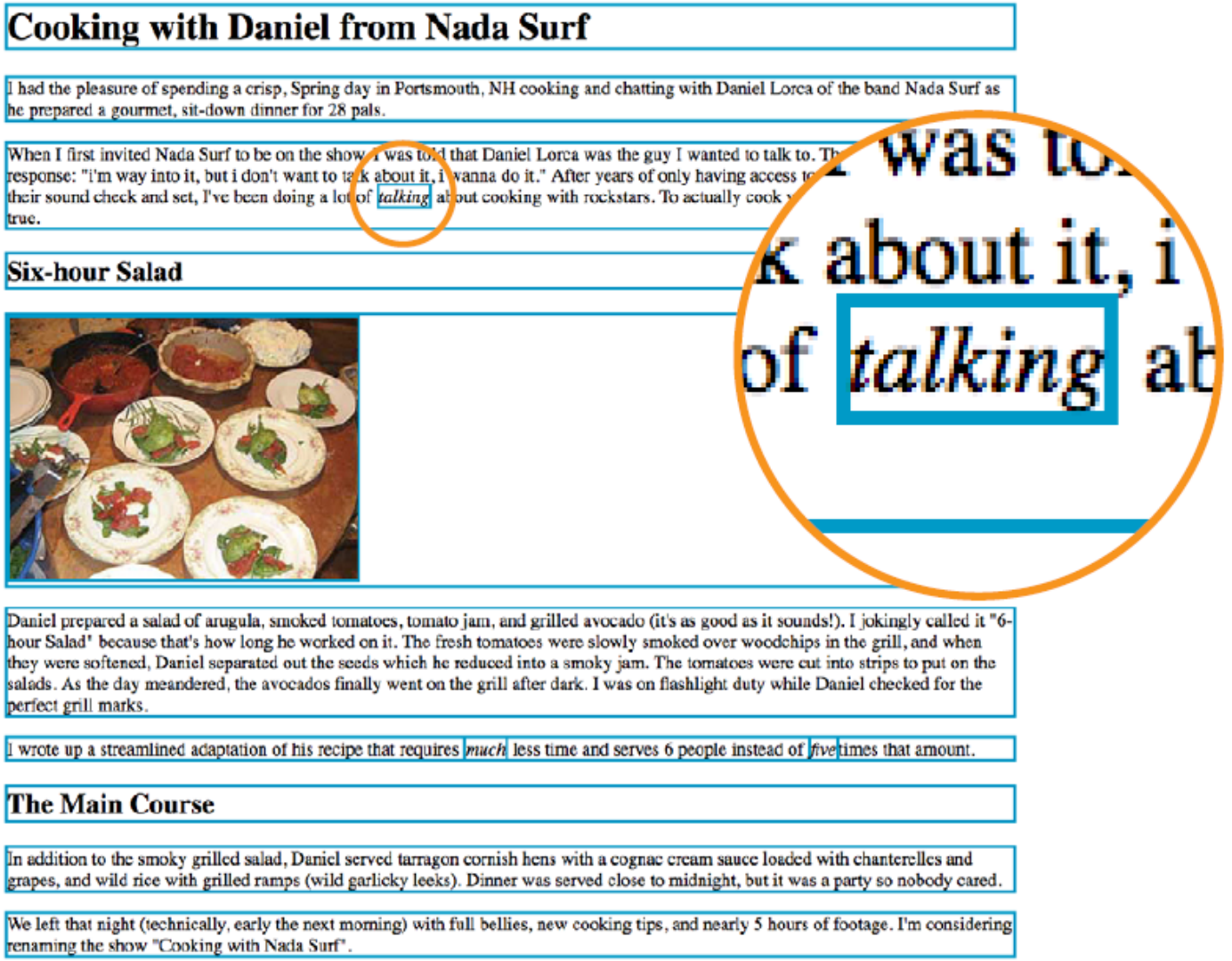


FIGURE 11-9. Rules around all the elements reveal their element boxes.

CSS Units of Measurements

CSS Units

CSS3 provides a variety of units of measurement. They fall into two broad categories: absolute and relative.

Absolute units

Absolute units have predefined meanings or real-world equivalents. With the exception of pixels, they are not appropriate for web pages that appear on screens.

px	pixel, defined as equal to 1/96 of an inch in CSS3.
in	inches.
mm	millimeters.
cm	centimeters.
q	¼ millimeter.
pt	points (1/72 inch). Points are a unit commonly used in print design.
pc	picas (1 pica = 12 points or 1/6 inch). Points are a unit commonly used in print design.

Relative units

Relative units are based on the size of something else, such as the default text size or the size of the parent element.

em	a unit of measurement equal to the current font size.
ex	x-height, approximately the height of a lowercase "x" in the font.
rem	root em, equal to the em size of the root element (**html**).
ch	zero width, equal to the width of a zero (0) in the current font and size.
vw	viewport width unit, equal to 1/100 of the current viewport (browser window) width.
vh	viewport height unit, equal to 1/100 of the current viewport height.
vmin	viewport minimum unit, equal to the value of **vw** or **vh**, whichever is smaller.
vmax	viewport maximum unit, equal to the value of **vw** or **vh**, whichever is larger.

NOTES

- Although not a "unit," percentages are another common measurement value for web page elements. Percentages are calculated relative to another value, such as the value of a property applied to the current element or its parent or ancestor. The spec always says what a percentage value for a property is calculated on.

  When used for page layouts, percentage values ensure that page elements stay proportional.
- Child elements do not inherit the relative values of their parent, but rather the resulting *calculated* value.
- IE9 supports **vm** instead of **vmin**. IE and Edge (all versions as of 2017) do not support **vmax**.

Absolute Units

- Absolute units have predefined meanings or real-world equivalents. They are always the same size, regardless of the context in which they appear.
- The most popular absolute unit for web design is the **pixel**, which CSS3 defines as 1/96 inch. Pixels are right at home on a pixel-based screen and offer precise control over the size of the text and elements on the page. For a while there, pixels were all we used. Then we realized they are too rigid for pages that need to adapt to a wide variety of screen sizes and user preferences.
- Relative measurements like rem, em, and % are more appropriate to the fluid nature of the medium.
- As long as we are kicking **px** to the curb, all of the absolute units—such as **pt**, **pc**, **in**, **mm**, and **cm**—are out because they are irrelevant on screens, although they may be useful for print style sheets. That narrows down your unit choices a bit.

Relative Units

- As I just established, relative units are the way to go for most web measurements, and there are a few options: **rem**, **em**, and **vw/vh**.

Relative Units

The rem unit

- CSS3 introduced a relative measurement called a **rem** (for **root em**) that is based on the font size of the **root (html)** element, whatever that happens to be. In modern browsers, the default root font size is **16** pixels; therefore, a rem is equivalent to a **16-pixel** unit (unless you set it explicitly to another value).
- An element sized to 10rem would measure 160 pixels.
- For the most part, you can use rem units like an absolute measurement in style rules; however, because it is relative, if the base font size changes, so does the size of a rem. If a user changes the base font size to 24 pixels for easier reading from a distance, or if the page is displayed on a device that has a default font size of 24 pixels, that 10rem element becomes 240 pixels. That seems dodgy, but rest assured that it is a feature, not a bug. There are many instances in which you want a layout element to expand should the text size increase. It keeps the page proportional with the font size, which can help maintain optimum line lengths.

Relative Units

The em unit

- An em is a relative unit of measurement that, in traditional typography, is based on the width of the capital letter M (thus the name “em”).
- In the CSS specification, an em is calculated as the distance between baselines when the font is set without any extra space between the lines (also known as leading).
- For text with a font size of 16 pixels, an em measures 16 pixels; for 12-pixel text, an em equals 12 pixels

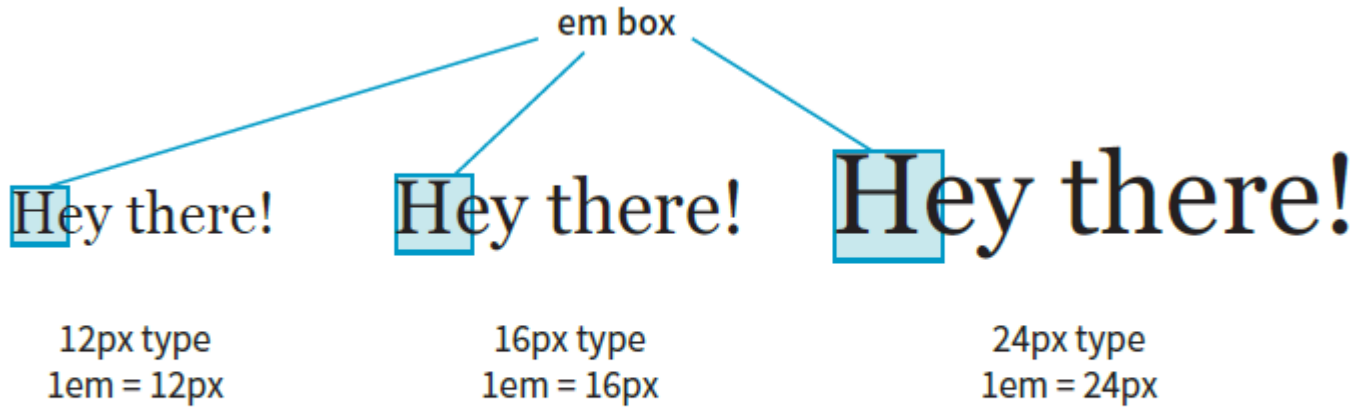


FIGURE 11-10. An em is based on the size of the text.

Relative Units

The em unit

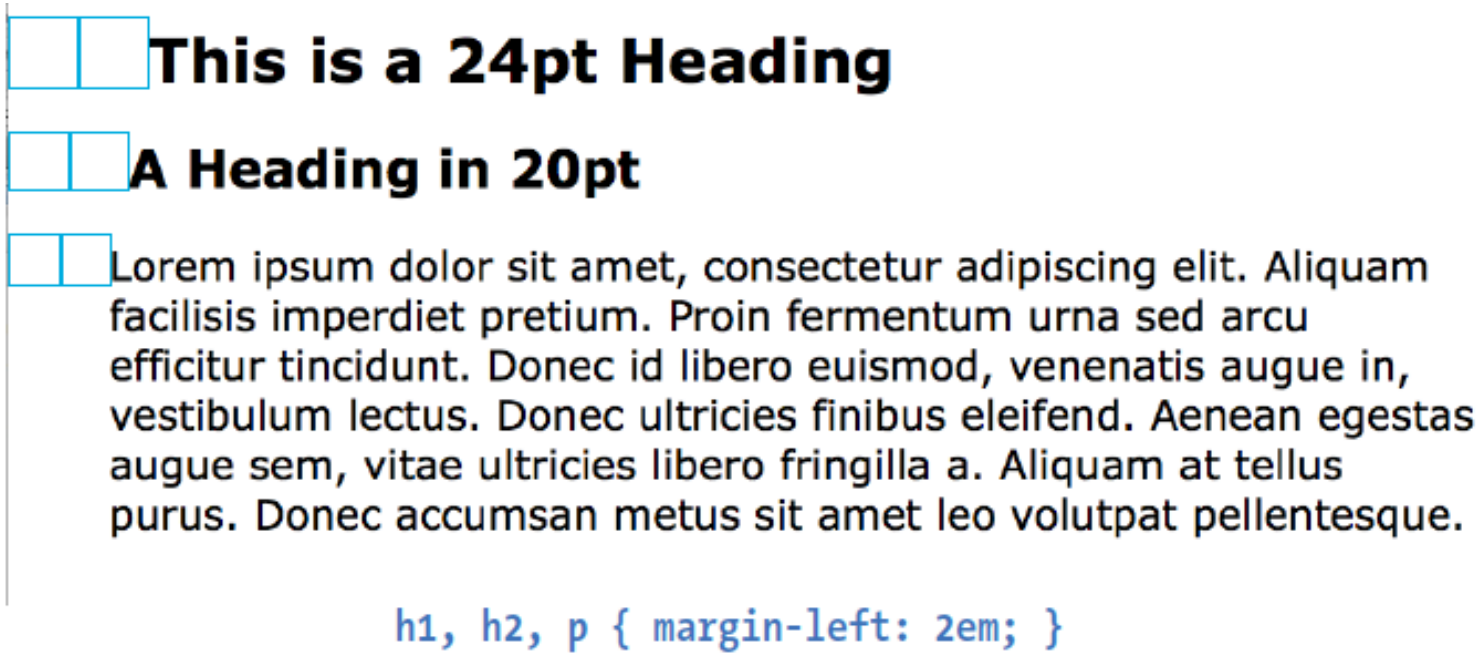


FIGURE 11-11. Em measurements are always relevant to the element's font size. An em for one element may not be the same for another.

Viewport percentage lengths (vw/vh)

vw and vh

Viewport width and view port height

- The viewport width (vw) and viewport height (vh) units are relative to the size of the viewport (browser window). A vw is equal to 1/100 the width of the viewport. Similarly, a vh is equal to 1/100 the height of the viewport.
- Viewport-based units are useful for making images and text elements stay the full width or height of the viewport:

```
header {
  width: 100vw;
  height: 100vh;
}
```

- It's also easy to specify a unit to be a specific percentage of the window size, such as 50%:

```
img {
  width: 50vw;
  height: 50vh;
}
```

- Related are the **vmin** unit (equal to the value of **vw** or **vh**, whichever is smaller) and **vmax** (equal to the value of **vw** or **vh**, whichever is larger).

CSS Selectors in Details

SECTION 5

Basic Selectors

Universal Selector:

```
* {    border: 1px solid #02C2D8;}
```

Element Selector:

```
li {    border: 2px solid #04B1D9;}
```

Class Selector:

```
.must-have {    border: 3px solid #0587BF;}
```

ID Selector:

```
#wants {    border: 4px solid #0668A4;}
```

Grouping CSS Selector:

```
#content, #needs, p {   border: 2px solid #20CC80; /*green*/
.want-to-have, #wants {    border: 2px solid #3D4173; /*purple*/}
```

Advanced Selectors

Example used for Advanced Selectors

```
<div id="content">
    <ul id="needs" class="needs">
            <li id="one" class="must-have">House</li>
            <li id="two" class="must-have">Land</li>
            <li id="three" class="must-have">Car</li>
    </ul>
    <div>
            <p class="qoute">Some essentials in life.</p>
            <div>
                <ul id="wants">
                        <li class="want-to-have">Mansion</li>
                        <li class="want-to-have">Ferrari</li>
                        <li class="want-to-have">Kingdom</li>
                </ul>
                <p class="qoute">Teach a man to fish and that makes you an awesome person.</p>
            </div>
    </div>
</div>
```

Graphical Representation of Element Descendants

```
<div id="qoute">
    <ul id="needs" class="needs">
        <li class="must-have">House</li>
        <li class="must-have">Land</li>
        <li class="must-have">Car</li>
    </ul>
</div>
```

I am a descentant of

We are descentants of
<ul id="needs" class="needs">

Indentation is a simple way to identify descendants of an element. The **<ul>** and **<li>** elements are descendants of the ** **.

Hint : Indent your code.

Closed

⊞

Opened

⊟

⊟

⊞

⊞

⊞

This representation shows all the elements that are contained inside another element are the descendants. **Descendants coloured orange.**

To Set Style for all descendants of ul id="needs" and including

- The syntax for a descendant combinator selector is the parent or ancestor followed by a whitespace then the descendant.

```
#needs li {
    border: 1px solid #02C2D8;
}
```

- Only the <li> elements under #needs

Graphical Representation of Child Elements

```
<div id="qoute">
    <ul id="needs" class="needs">
        <li class="must-have">House</li>
        <li class="must-have">Land</li>
        <li class="must-have">Car</li>
    </ul>
</div>
```

I am a child of

We are children of
<ul id="needs" class="needs">

The child has a direct relationship with its parent. The ** ** has a **<ul>** as a child and the **<ul>** has **<li>** elements as children but the **<li>** elements are not direct children of ** **.

Closed

⊞

Opened

⊟

⊟

⊞

⊞

⊞

The Lone Child

This story follows the journey of a ** ** who has only one **purple** child named **<ul>**. Though **<ul>** has children of his own, they are not children of ** **.

To set the Style for all of the Children for

- The syntax for a child combinator selector is the parent followed by a greater than (>) sign then the child.

```
#quote > ul {
    border: 2px solid #04B1D9;
}
```

- In this example, it only has a child <ul>, but it can be of any type of elements such as <textarea> and etc.

Graphical Representation of Sibling Elements

Adjacent Siblings

```
<div id="qoute">
    <ul id="needs" class="needs">
        <li id="one">House</li>
        <li id="two">Land</li>
        <li id="three">Car</li>
    </ul>
</div>
```

House and *Land* are Adjacent Siblings

Land and *Car* are Adjacent Siblings

Adjacent Siblings

Closed

⊞

Opened

⊟

⊟

⊞

⊞

⊞

Caring Siblings

Three **<li>** siblings. Red, Green and Blue. They all together love each other, especially Green it's true.

Red is adjacent to Green and Green is adjacent to Blue.

Forever and ever they stay together even though Red's not adjacent to Blue.

Set the Style for the next Sibling for <li id="one">

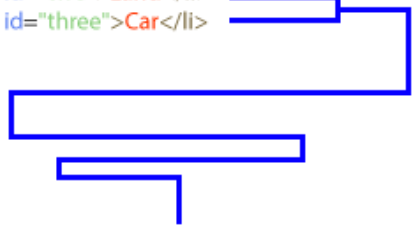
- The syntax for the adjacent sibling combinator is the first element in the selector sequence followed by the plus (+) sign then the second element that immediately succeeds the first.

```
#one + li {
    border: 3px solid #06BD06;
}
```

Graphical Representation of Sibling Elements

General Siblings

```
<div id="qoute">
    <ul id="needs" class="needs">
        <li id="one">House</li> ★
        <li id="two">Land</li>
        <li id="three">Car</li>
    </ul>
</div>
```



Land and *Car* are the general siblings of *House*.

The star (★) represents the element whose relationship is being described.
i.e The <li> containing text "House".

Set the Style of all of the Siblings of <li id="one">

- The syntax for the general sibling combinator is the first element in the selector sequence followed by the tilde (~) sign then a simple selector. i.e type, class, id etc.

```
#one ~ li {
    border: 3px solid #0F860F;
}
```

Making a Long Selector

- Here we will make a long and valid selector to see how much you have grasped. So far when using a class selector, every element with that class is matched. This is because we have not been specifying where in the DOM tree to look, so all is matched.

```
#quote > ul  ul > #one ~ li {
    border: 3px solid #FF6EC7;
}
```

Summery for CSS Selectors

- Combinators gives us the ability to go crazy with our selectors. Selectors can become long and complicated; it is good to practice using combinators to make lengthy selectors. This will show that you have a thorough understanding of what has been covered.

A Word About Property Listings

Each CSS property listing in this book is accompanied by information on how it behaves and how to use it. Property listings include:

Values:

These are the accepted values for the property. Predefined keyword values appear
in code font (for example, **small**, **italic**, or **small-caps**) and must be typed in
exactly as shown.

Default:

This is the value that will be used for the property by default (its initial value)— that is, if no other value is specified. Note that the default browser style sheet values may vary from the defaults defined in CSS.

Applies to:

Some properties apply only to certain types of elements.

Inherits:

This indicates whether the property is passed down to the element's descendants

CSS-wide keywords

All CSS properties accept the three CSS-wide keywords: **initial**, **inherit**, and **unset**. Because they are shared by all properties, they are not listed with the values for individual property listings.

- The **initial** keyword explicitly sets the property to its default (initial) value.
- The **inherit** keyword allows you to explicitly force an element to inherit a style property from its parent. This may come in handy to override other styles applied to that element and to guarantee that the element always matches its parent.
- Finally, **unset** erases declared values occurring earlier in the cascade, setting the property to either **inherit** or **initial**, depending on whether it inherits or not.

Text

SECTION 6

Font Properties

The CSS2.1 font-related properties are universally supported:

- `font-family`
- `font-size`
- `font-weight`
- `font-style`
- `font-variant`
- `font`

Font Properties

The CSS Font Module Level 3 adds these properties for more sophisticated font handling, although browser support is inconsistent as of this writing:

```
font-stretch
font-variant-ligatures
font-variant-position
font-variant-caps
font-variant-numeric
font-variant-alternates
font-variant-east-asian
font-size-adjust
font-kerning
font-feature-settings
font-language-override
```

Formatting Text
(Attributes of Text)

Font: family, size, weight, style, variant

Color

Line Adjustment: height, indents, align

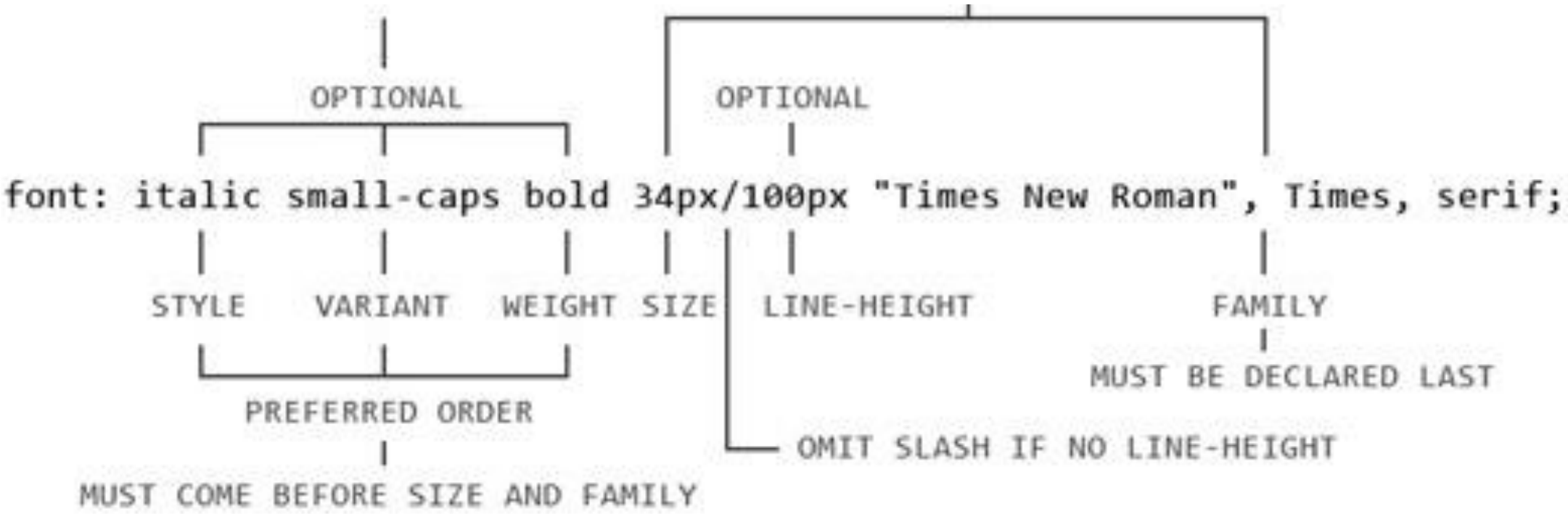
Text-decoration

Capitalization

Spacing: line-spacing, word-spacing

Shadow

Bullets and Buttons: list-style-type, list-style-position, list-style-image



Font Properties

Specifying the Font Name

font-family

Values:	*one or more font or generic font family names, separated by commas*
Default:	depends on the browser
Applies to:	all elements
Inherits:	yes

- Use the **font-family** property to specify a font or list of fonts (known as a font stack) by name, as shown in these examples:

```
body { font-family: Arial; }
var { font-family: Courier,
monospace; }
p { font-family: "Duru Sans",
     Verdana, sans-serif; }
```

Specifying the Font Name

Here are some important syntax requirements:

- All font names, with the exception of generic font families, must be capitalized.
- For example, use **Arial** instead of **arial**.
- Use commas to separate multiple font names, as shown in the second and third examples.
- Notice that font names that contain a character space (such as Duru Sans in the third example) must appear within quotation marks.

Generic font families

Font-family: font|initial|inherit;

FIGURE 12-2. Examples of the five generic font families.

serif

Latin fonts	Times New Roman, Bodoni, Garamond, Minion Web, ITC Stone Serif, MS Georgia, Bitstream Cyberbit
Greek fonts	Bitstream Cyberbit
Cyrillic fonts	Adobe Minion Cyrillic, Excelcior Cyrillic Upright, Monotype Albion 70, Bitstream Cyberbit, ER Bukinst
Hebrew fonts	New Peninim, Raanana, Bitstream Cyberbit
Japanese fonts	Ryumin Light-KL, Kyokasho ICA, Futo Min A101
Arabic fonts	Bitstream Cyberbit
Cherokee fonts	Lo Cicero Cherokee

sans-serif

Latin fonts	MS Trebuchet, ITC Avant Garde Gothic, MS Arial, MS Verdana, Univers, Futura, ITC Stone Sans, Gill Sans, Akzidenz Grotesk, Helvetica
Greek fonts	Attika, Typiko New Era, MS Tahoma, Monotype Gill Sans 571, Helvetica Greek
Cyrillic fonts	Helvetica Cyrillic, ER Univers, Lucida Sans Unicode, Bastion
Hebrew fonts	Arial Hebrew, MS Tahoma
Japanese fonts	Shin Go, Heisei Kaku Gothic W5
Arabic fonts	MS Tahoma

cursive

Latin fonts	Caflisch Script, Adobe Poetica, Sanvito, Ex Ponto, Snell Roundhand, Zapf-Chancery
Cyrillic fonts	ER Architekt
Hebrew fonts	Corsiva
Arabic fonts	DecoType Naskh, Monotype Urdu 507

fantasy

Latin fonts	Alpha Geometrique, Critter, Cottonwood, FB Reactor, Studz

monospace

Latin fonts	Courier, MS Courier New, Prestige, Everson Mono
Greek Fonts	MS Courier New, Everson Mono
Cyrillic fonts	ER Kurier, Everson Mono
Japanese fonts	Osaka Monospaced
Cherokee fonts	Everson Mono

```
<head>
    <meta charset="utf-8">
    <title>Black Goose Bistro
            Summer Menu</title>
    <link
   href="http://fonts.googleapis.com/css?f
   amily=Marko+One" rel="stylesheet">
    <style>
      body { font-family:Verdana, sans-
             serif;}
      h1 {font-family: "Marko One",
          Georgia, serif;}}
    </style>
</head>
```

font-family:

font-family: special font, font-family, generic-family

font-size

Values: *length unit* | *percentage* | xx-small | x-small | small | medium | large | x-large | xx-large | smaller | larger

Default: medium

Applies to: all elements

Inherits: yes

CSS Units Cheat Sheet

As a quick reference, here are the CSS length units again:

Relative units

em ex rem ch

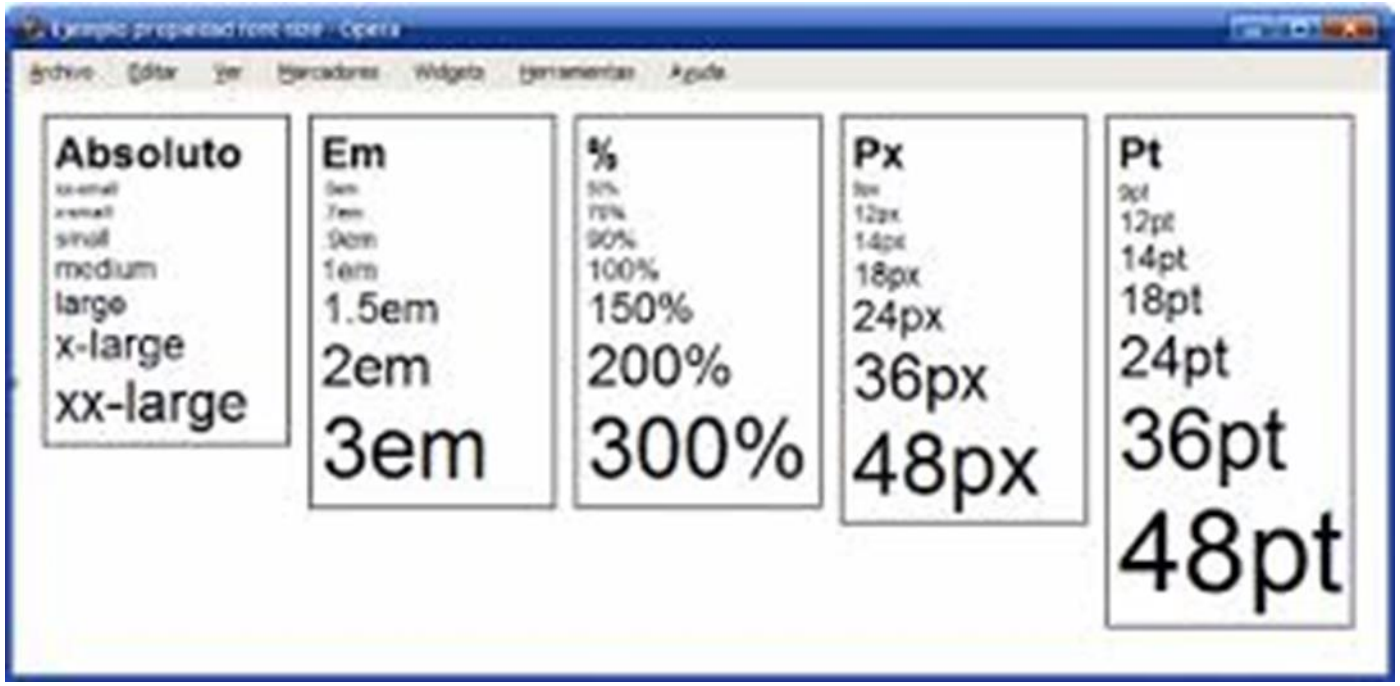
vw vh vmin vmax

Absolute units

px in mm cm

q pt pc

Specifying Font Size



Specifying Font Size

Font-size: absolute | em | % | px | pt

THE MARKUP

```
<h1>Headline in Body</h1>
<p>Pellentesque ligula leo,…</p>
<article>
  <h1>Headline in Article</h1>
  <p>Vivamus …</p>
</article>
```

THE STYLES

```
h1 {
  font-size: 1.5em;    /* sets all h1s to 1.5em */
}
article {
  font-size: .875em    /* 14 pixels based on 16px default */
}
```

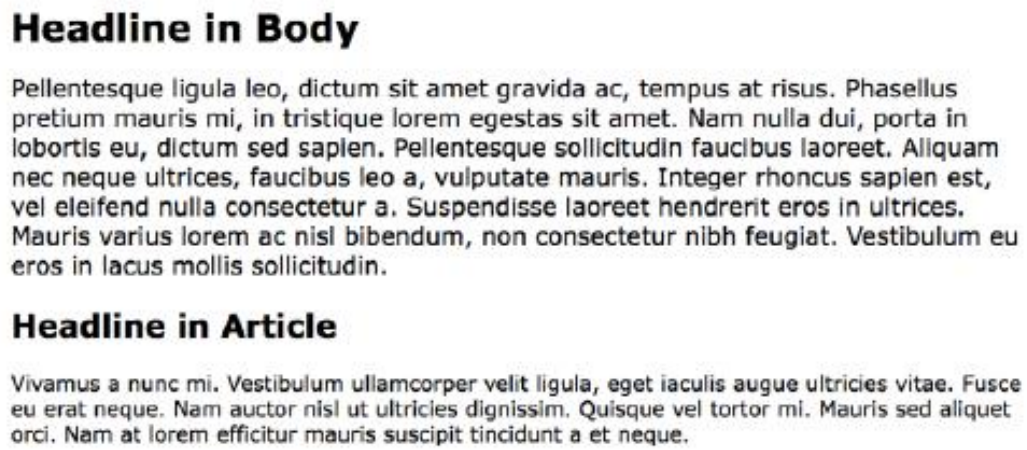


FIGURE 12-4. All **h1** elements are sized at 1.5em, but they are different sizes because of the context in which they appear.

Specifying Font Size

Font-size: absolute | em | % | px | pt

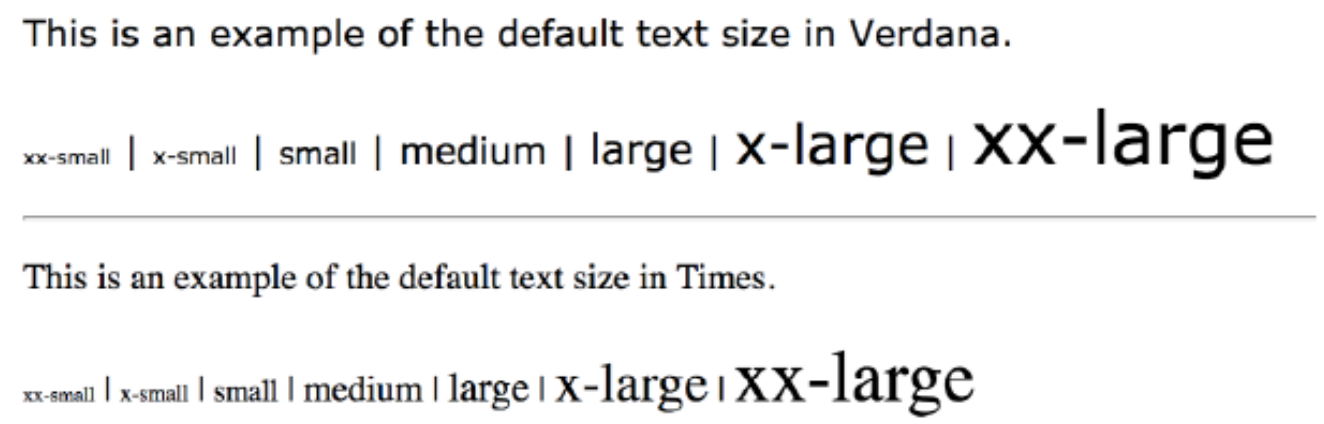


FIGURE 12-5. Text sized with absolute keywords.

Specifying Font Size

Font-size: absolute | em | % | px | pt

100 Thin
200 Light
300 Book
400 Regular
500 Medium
600 DemiBold
700 Bold
800 ExtraBold
900 Heavy

The font weight is lighter.

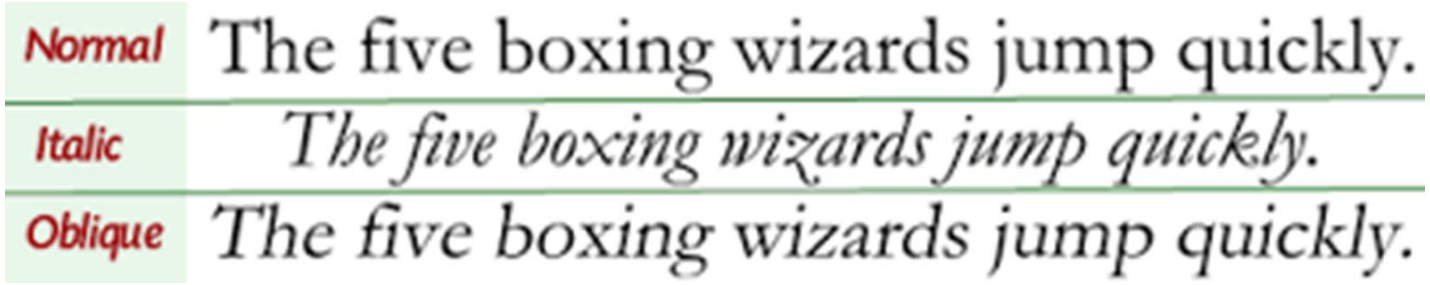
The font weight is normal.

The font weight is bold.

The font weight is bolder.

Font-Weight

font-weight: normal | bold | bolder | lighter | 100 | 200 | 300 | 400 | 500 | 600 | 700 | 800 | 900 | inherit



Font Style

Font-style: normal | italic | oblique | inherit

font-variant:

font-variant: normal | small-caps | inherit

font-stretch

Values:	normal | ultra-condensed | extra-condensed | condensed | semi-condensed | semi-expanded | expanded | extra-expanded | ultra-expanded
Default:	normal
Applies to:	all elements
Inherits:	yes

Font Stretch (Condensed and Extended)

FIGURE 12-9. Examples of condensed, normal, and extended versions of the Universe typeface.

Font Stretch (Condensed and Extended)

Text Style

SECTION 7

color

Values:	*color value (name or numeric)*
Default:	depends on the browser and user's preferences
Applies to:	all elements
Inherits:	yes

Changing Text Color

Changing Text Color

Set by Color Name:

```
h1 { color:gray; }
```

Set by RBG Code:

```
h1 { color:rgb(102, 102, 102) }
```

Set by RGB code in Hexadecimal Format:

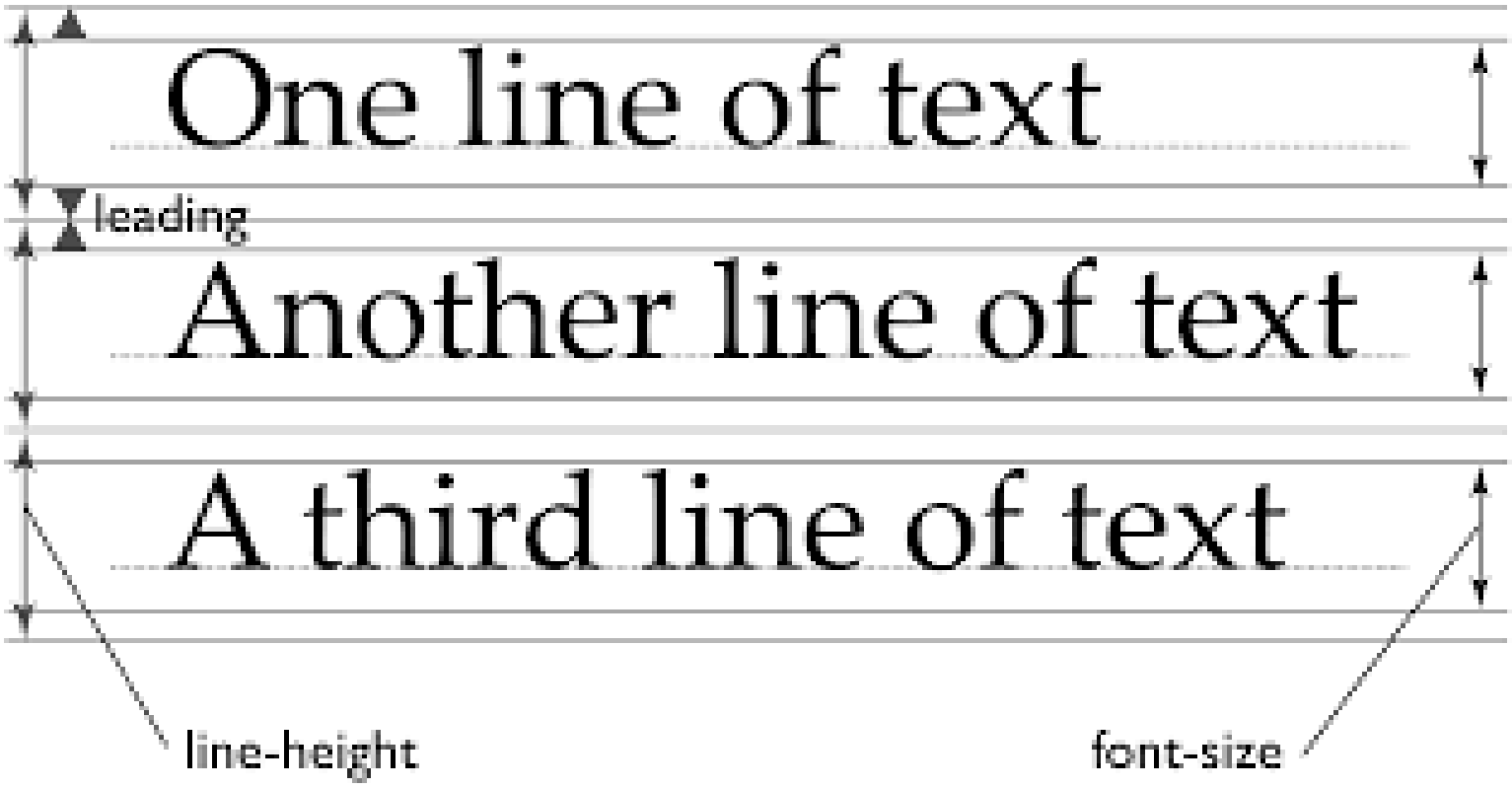
```
h1 { color:#666666; }
```

Set by Condensed RGB code:

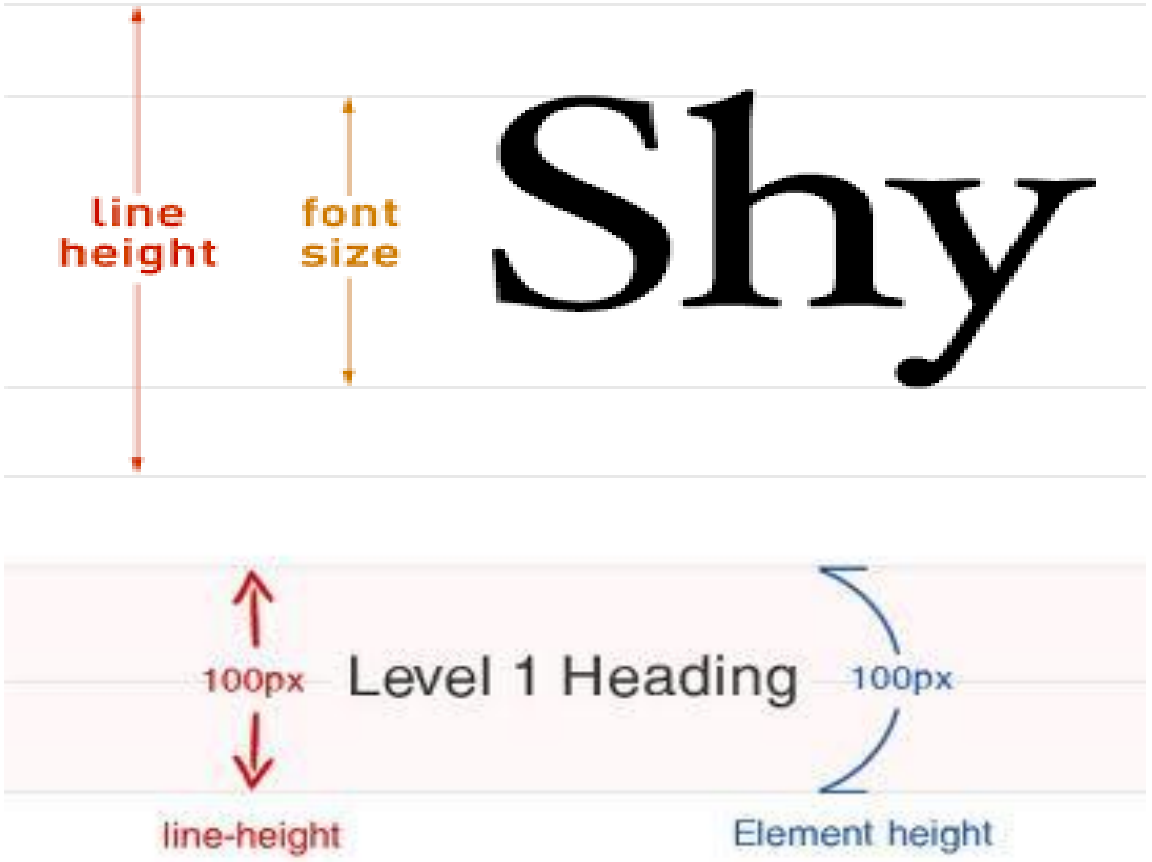
```
h1 { color:#FC0; }   ==  h1 { color:#FFCC00; }
```

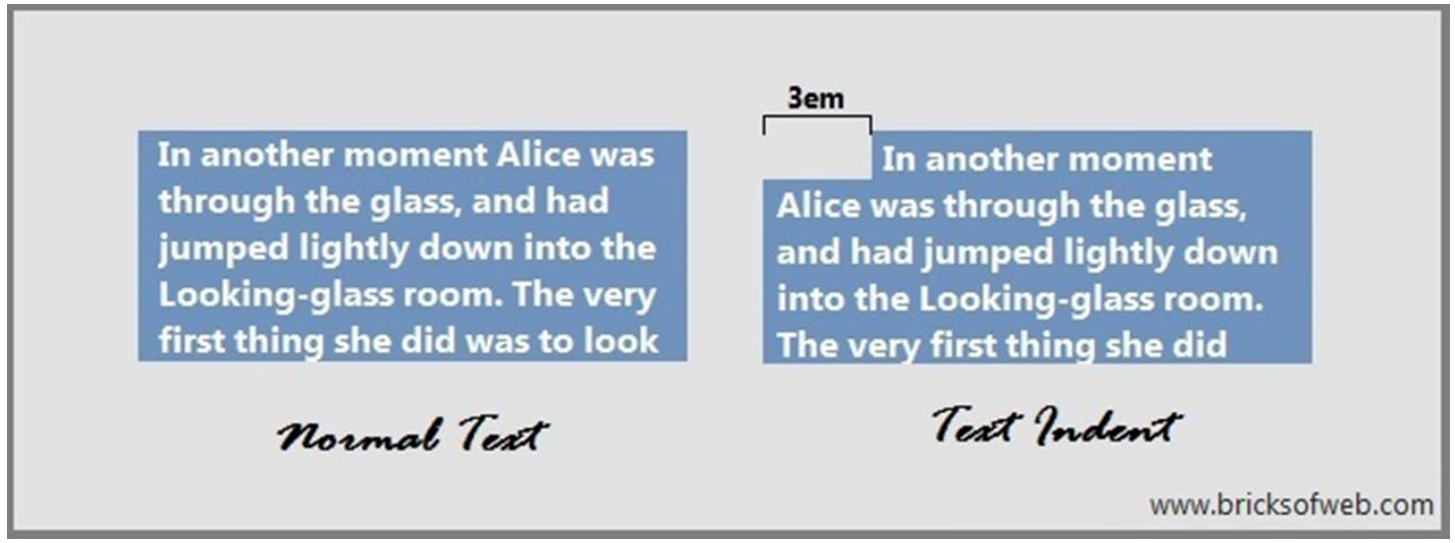
Color name will be covered in the next chapter.

Text Line Adjustment



Text Line Adjustment





Text Indentation

Text-indent: length_measurement | percentage | inherit

```
p#1 {text-indent: 2em;}
p#2 {text-indent: 25%;}
p#3 {text-indent: -35px;}
```

paragraph 1 is indented for 2em

25% paragraph is indented for 25%

Paragraph 3 is indented right -35px here for -35px

text-align: left;

Paragraph 1. The text-align property controls the horizontal alignment of the text within an element. It does not affect the alignment of the element on the page. The resulting text behavior of the various values should be fairly intuitive.

text-align: right;

Paragraph 2. The text-align property controls the horizontal alignment of the text within an element. It does not affect the alignment of the element on the page.The resulting text behavior of the various values should be fairly intuitive.

text-align: center;

Paragraph 3. The text-align property controls the horizontal alignment of the text within an element. It does not affect the alignment of the element on the page.The resulting text behavior of the various values should be fairly intuitive.

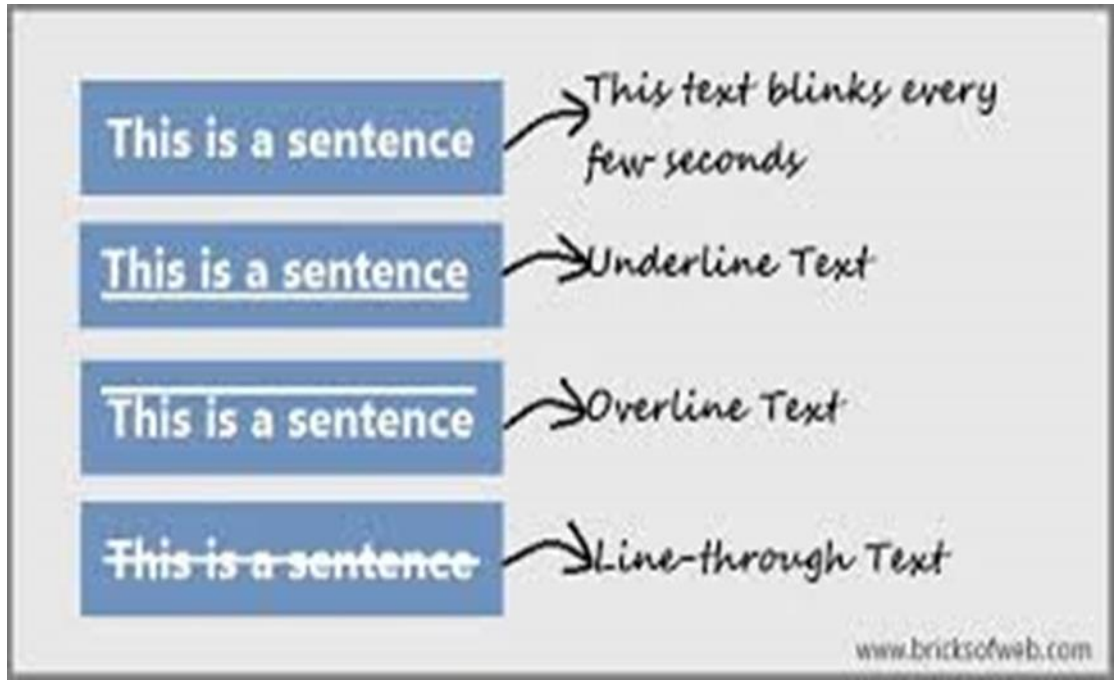
text-align: justify;

Paragraph 4. The text-align property controls the horizontal alignment of the text within an element. It does not affect the alignment of the element on the page.The resulting text behavior of the various values should be fairly intuitive.

FIGURE 12-14. Examples of CSS2.1 **text-align** values.

Horizontal Alignment

text-align: right | left | center | justify | inherit



Underlines and Decorations

text-decoration:line-through

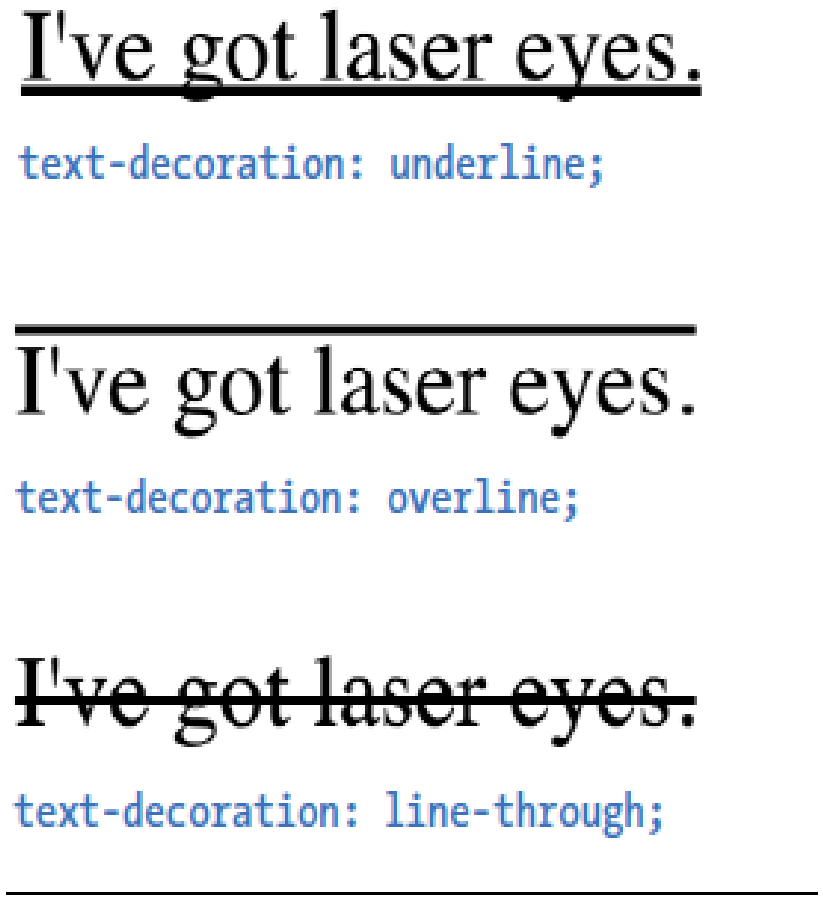
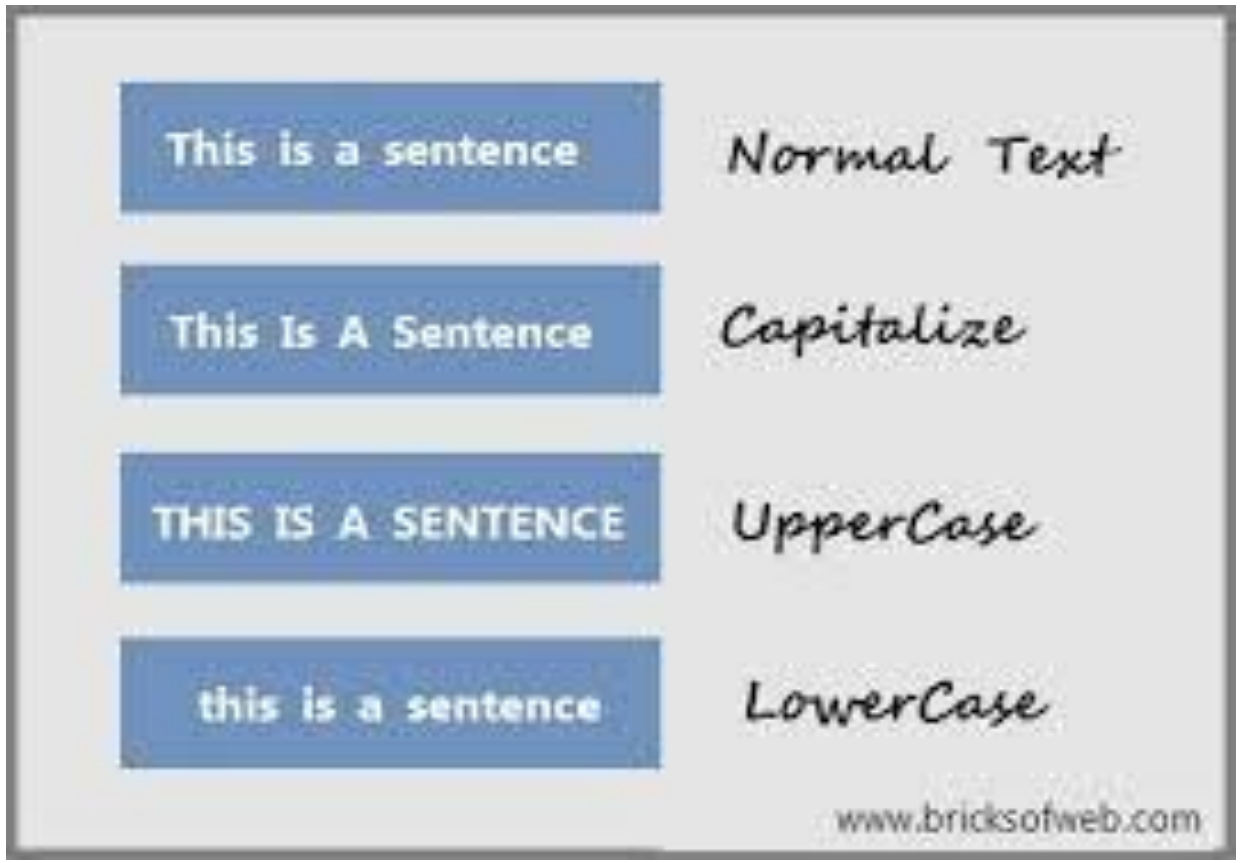


FIGURE 12-15. Examples of **text-decoration** values.

Underlines and Decorations

text-decoration:line-through



Changing Capitalization

text-transform:none | capitalize | lowercase | uppercase | inherit

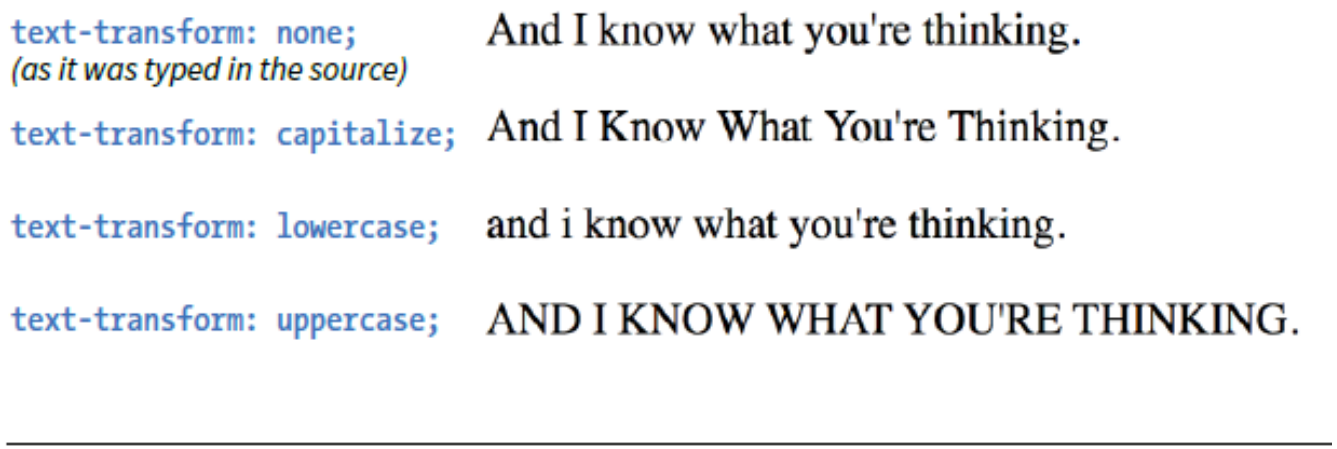
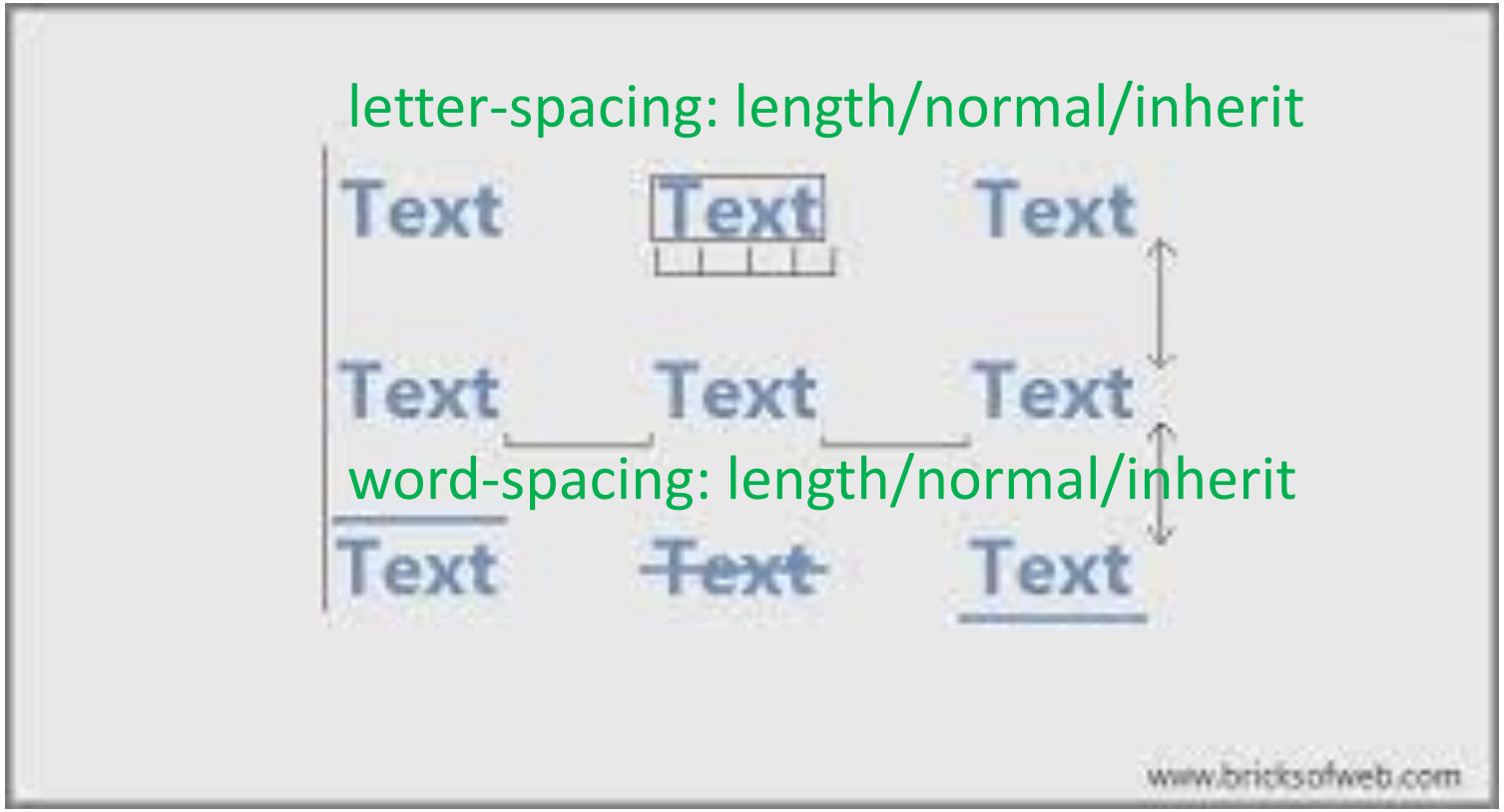


FIGURE 12-16. The **text-transform** property changes the capitalization of characters when they are displayed, regardless of how they are typed in the source.

Changing Capitalization

text-transform:none | capitalize | lowercase | uppercase | inherit



Text Spacing

B l a c k G o o s e B i s t r o S u m m e r M e n u

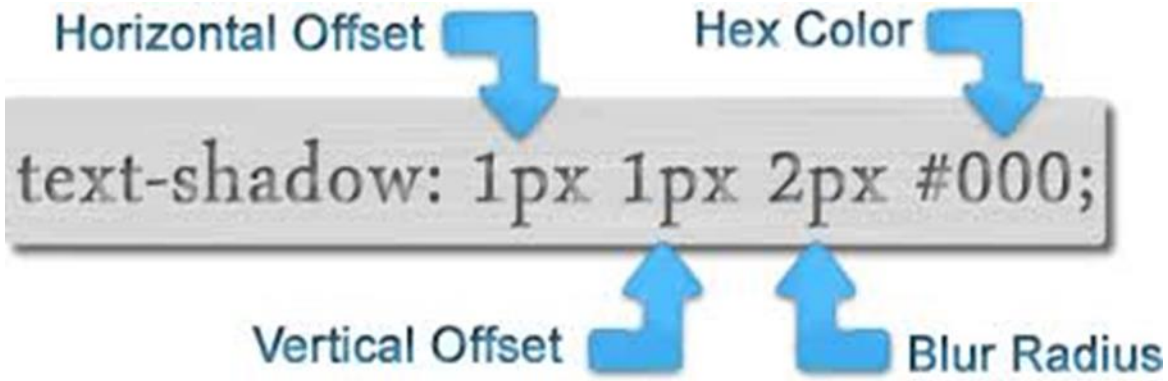
```
p { letter-spacing: 8px; }
```

Black Goose Bistro Summer Menu

```
p { word-spacing: 1.5em; }
```

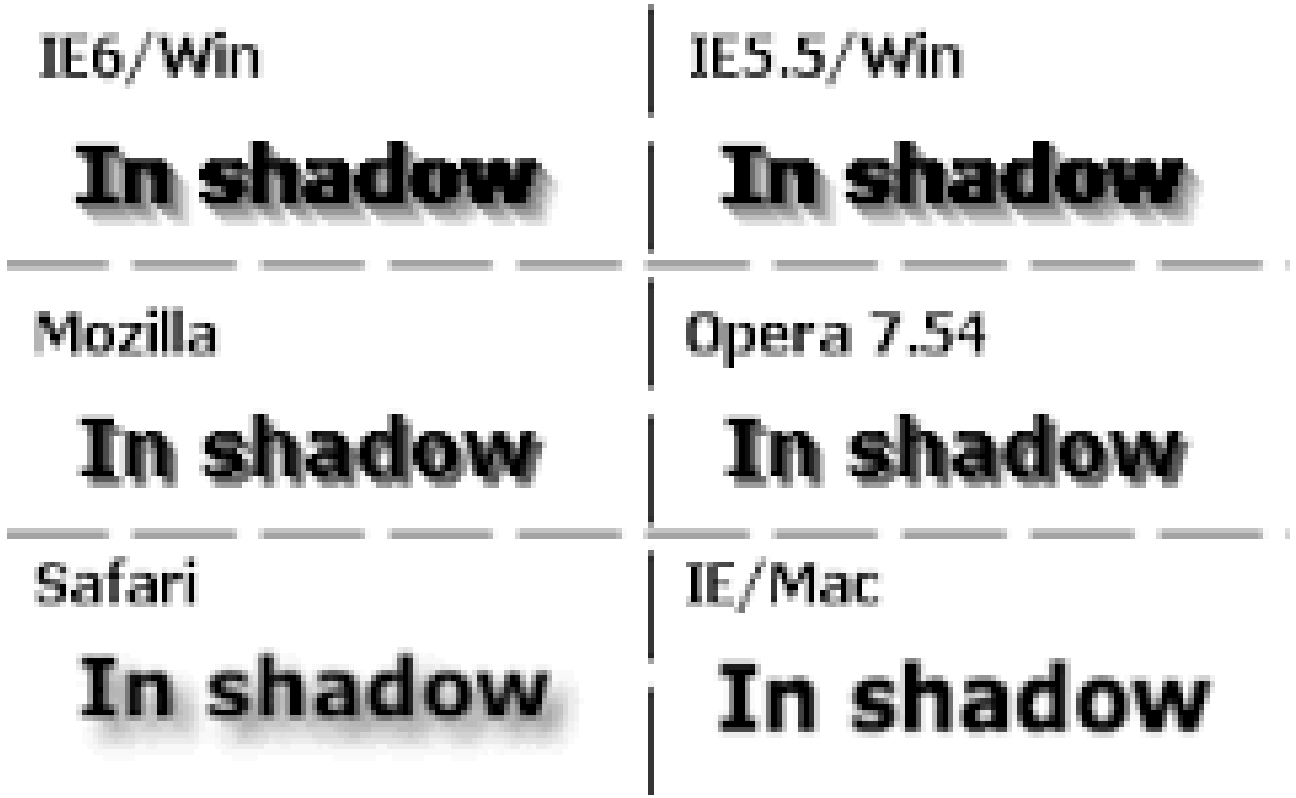
FIGURE 12-17. **letter-spacing** (top) and **word-spacing** (bottom).

Text Spacing



Text Shadow

text-shadow: ‘horizontal offset’ ‘vertical offset’ ‘blur radius’ ‘color’



Text Shadow

text-shadow: ‘horizontal offset’ ‘vertical offset’ ‘blur radius’ ‘color’

FIGURE 12-18. A minimal text drop shadow.

FIGURE 12-19. Adding a blur radius to a text drop shadow.

Text Shadow

text-shadow: 'horizontal offset' 'vertical offset' 'blur radius' 'color'

white-space (*CSS2*) Specifies how whitespace in the element source is handled in layout. For example, the **pre** value preserves the character spaces and returns found in the source, similar to the **pre** HTML element.

vertical-align (*CSS2*) Specifies the vertical alignment of an inline element's baseline relative to the baseline of the surrounding text. It is also used to set the vertical alignment of content in a table cell (**td**).

word-break *and* **line-break** (*CSS3*) Affects how text wrapping is calculated within words and lines, respectively, in various languages, including East Asian (Chinese, Japanese, Korean).

text-justify (*CSS3*) Specifies the manner in which space is to be added within and between words when the **text-align** property on the element is set to **justify**.

text-align-last (*CSS3*) Specifies how the last line of a block of text should be justified when the **text-align** property on the element is set to **justify**. For example, it is often preferable to have the last line left-justified for justified text to avoid awkwardly spaced words.

tab-size (*CSS3*) Specifies the length of the tab character (Unicode point 0009) in number of characters or a length measurement.

hyphens (*CSS3*) Provides control over how text is hyphenated. **manual** means hyphenation happens only when there is a hyphen added in the source. **auto** gives control to the browser, and **none** turns off hyphenation completely.

overflow-wrap (*CSS3*) Specifies whether browsers are allowed to break words to fit text in its bounding box.

hanging-punctuation (*CSS3*) Determines whether the punctuation mark may be outside the element's line box at the start or end of a line. Hanging punctuation can make margins appear more tidy.

The following properties are in the spec, but should not be used. Use the **dir** HTML attribute instead.

direction (*CSS3*) Specifies the direction in which the text reads: left to right (**ltr**) or right to left (**rtl**).

unicode-bidi (*CSS2*) Related to bidirectional features of Unicode. The Recommendation states that it allows the author to generate levels of embedding within the Unicode embedding algorithm. If you have no idea what this means, don't worry. Neither do I.

The Other Text Properties

Bullets and Numbers

SECTION 8

Choosing a Marker

Apply the **list-style-type** property to the **ul**, **ol**, or **li** element select the type of marker that appears before each list item (see Note).

list-style-type

Values:	none | disc | circle | square | decimal | decimal-leading-zero | lower-alpha | upper-alpha | lower-latin | upper-latin | lower-roman |upper-roman | lower-greek
Default:	disc
Applies to:	**ul**, **ol**, and **li** (or elements whose display value is **list-item**)
Inherits:	yes

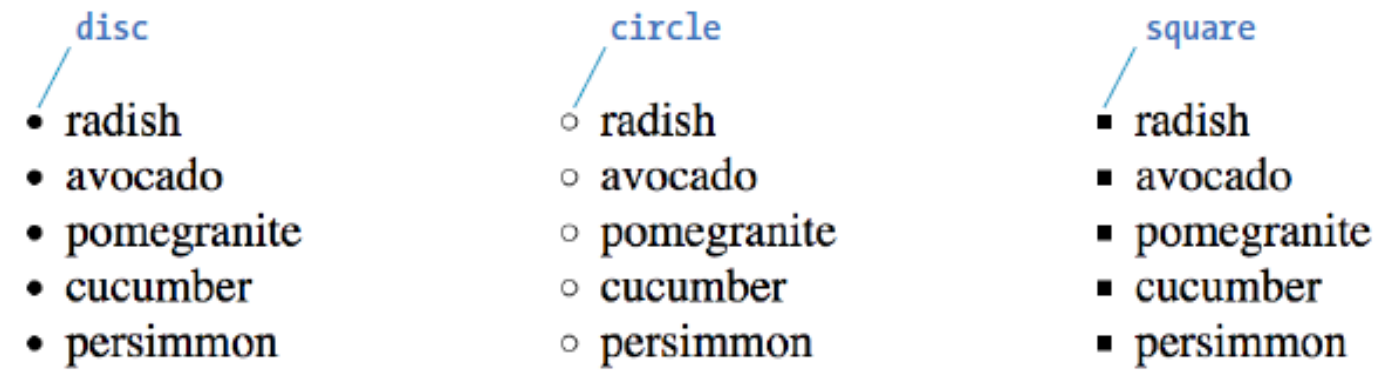


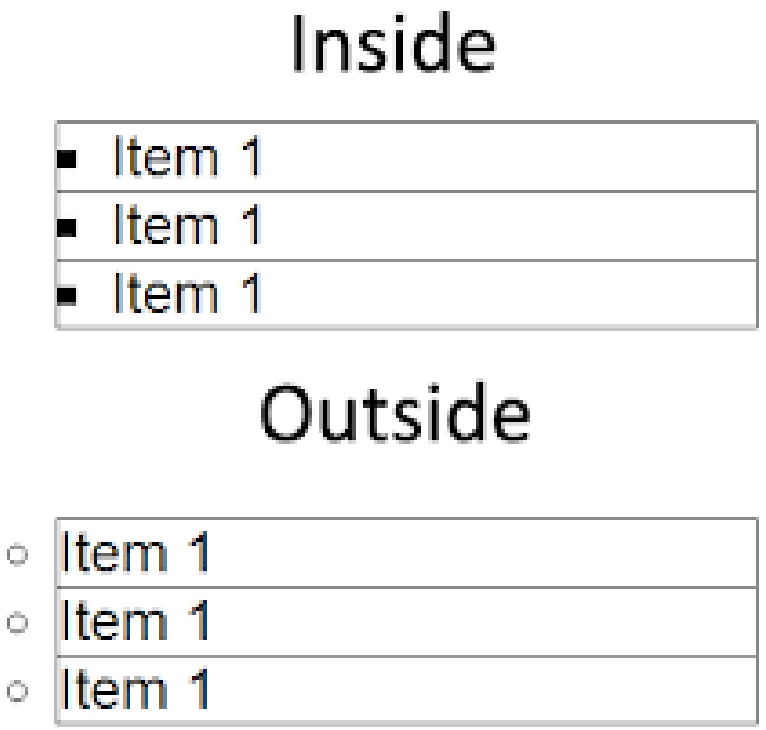
FIGURE 12-21. The `list-style-type` values `disc`, `circle`, and `square`.

Choosing a Marker

TABLE 12-1. Lettering and numbering system (CSS2.1)

Keyword	System
`decimal`	1, 2, 3, 4, 5...
`decimal-leading-zero`	01, 02, 03, 04, 05...
`lower-alpha`	a, b, c, d, e...
`upper-alpha`	A, B, C, D, E...
`lower-latin`	a, b, c, d, e... (same as lower-alpha)
`upper-latin`	A, B, C, D, E... (same as upper-alpha)
`lower-roman`	i, ii, iii, iv, v...
`upper-roman`	I, II, III, IV, V...
`lower-greek`	α, β, γ, δ, ε...

Lettering and numbering system



Bullets and Numbers

list-style-position: inside | outside | inherit

outside

- **Radish.** Praesent in lacinia risus. Morbi urna ipsum, efficitur id erat pellentesque, tincidunt commodo sem. Phasellus est velit, porttitor vel dignissim vitae, commodo ut urna.
- **Avocado.** Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos. Curabitur lacinia accumsan est, ut malesuada lorem consectetur eu.
- **Pomegranite.** Nam euismod a ligula ac bibendum. Aenean ac justo eget lorem dapibus aliquet. Vestibulum vitae luctus orci, id tincidunt nunc. In a mauris odio. Duis convallis enim nunc.

inside

- **Radish.** Praesent in lacinia risus. Morbi urna ipsum, efficitur id erat pellentesque, tincidunt commodo sem. Phasellus est velit, porttitor vel dignissim vitae, commodo ut urna.
- **Avocado.** Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos. Curabitur lacinia accumsan est, ut malesuada lorem consectetur eu.
- **Pomegranite.** Nam euismod a ligula ac bibendum. Aenean ac justo eget lorem dapibus aliquet. Vestibulum vitae luctus orci, id tincidunt nunc. In a mauris odio. Duis convallis enim nunc.

FIGURE 12-22. The **list-style-position** property.

Bullets and Numbers

list-style-position: inside | outside | inherit

You can also use your own image as a bullet by using the **list-style-image** property.

list-style-image

Values:	url(*location*) | none
Default:	none
Applies to:	**ul**, **ol**, and **li** (or elements whose display value is **list-item**)
Inherits:	yes

```
ul {
  list-style-type: disc;
  list-style-image: url(/images/rainbow.gif);
  list-style-position: outside;
}
```

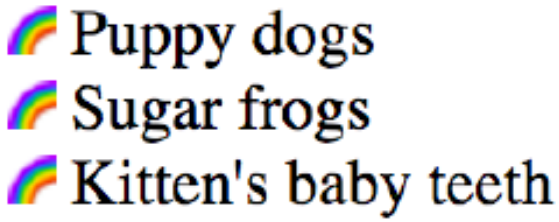


FIGURE 12-23. Using an image as a marker.

Make Your Own Bullets

Color

SECTION 9

Specify Color Values

- There are two main ways to specify colors in style sheets—with a predefined color name, as we have been doing so far:

  color: red; color: olive; color: blue;

- Or, more commonly, with a numeric value that describes a particular RGB color (the color model on computer monitors). You may have seen color values that look like these:

  color: #FF0000; color: #808000; color: #00F;

Color Names

```
color: silver;
background-color: gray;
border-bottom-color: teal;
```

FIGURE 13-1. The 17 standard color names in CSS2.1. (Note that "gray" must be spelled with an "a.")

RGB Color Values

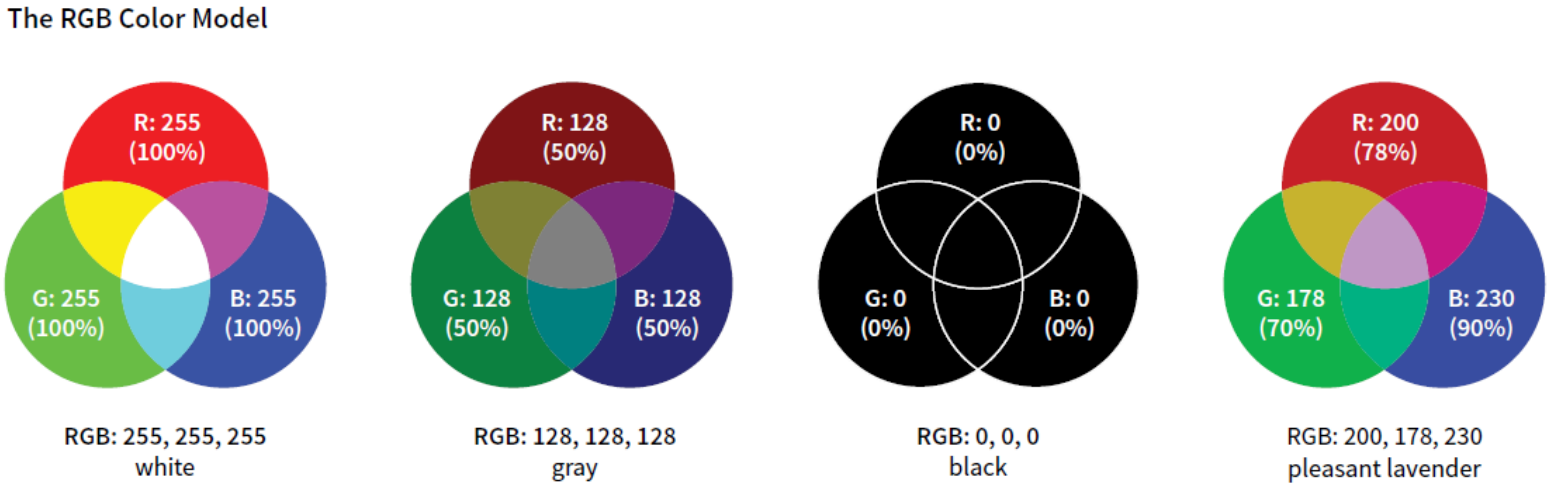
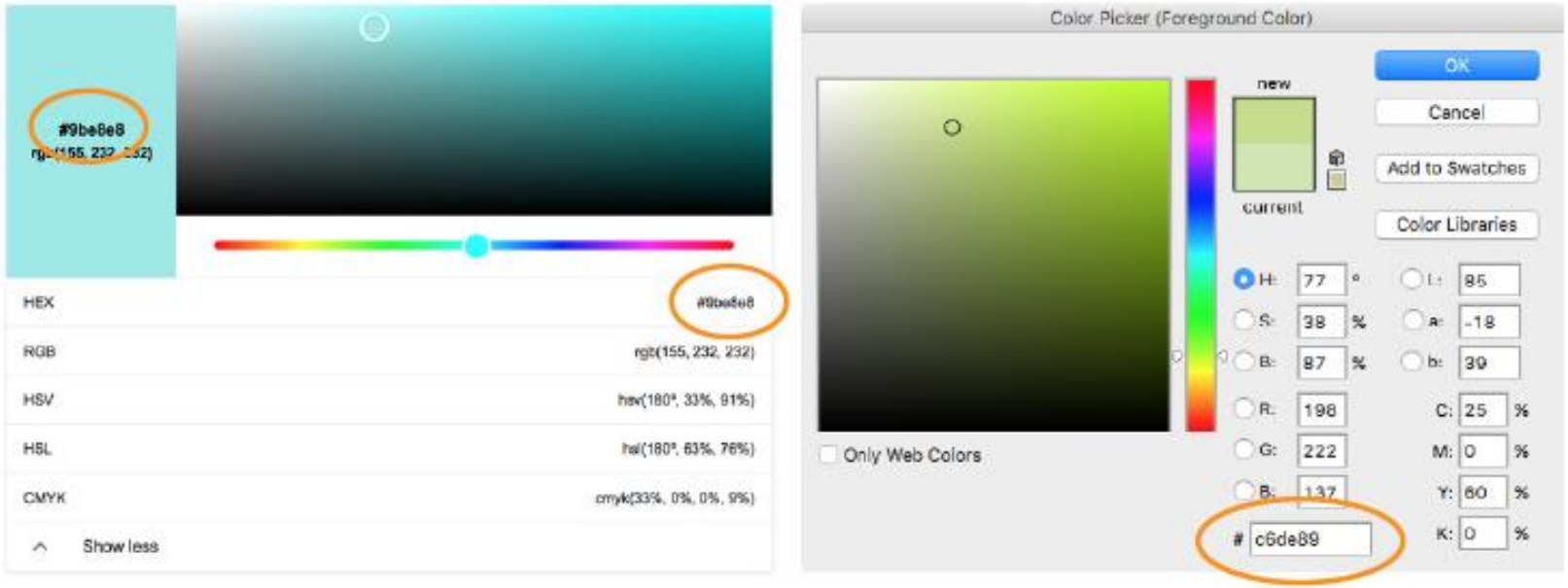


FIGURE 13-3. Computers create colors on a monitor by mixing different amounts of red, green, and blue light (thus, RGB). The color in the middle of each diagram shows what happens when the three color channels are combined. The more light there is in each channel (i.e., the higher the number value), the closer the combination is to white.

RGB Color Values

Color picker



Google color picker

Photoshop color picker

FIGURE 13-4. Color pickers such as the one at Google.com (search "color picker") and in Photoshop.

Writing RGB values in style sheets

color: rgb(200, 178, 230);
color: rgb(78%, 70%, 90%);
color: #C8B2E6;
color: #FC0; or color: #936;

- Specifying RGB Values
- There are four formats for providing

```
RGB values in CSS:
rgb(255, 255, 255)
rgb(100%, 100%, 100%)
#FFFFFF
#FFF
```

- All of these examples specify white.

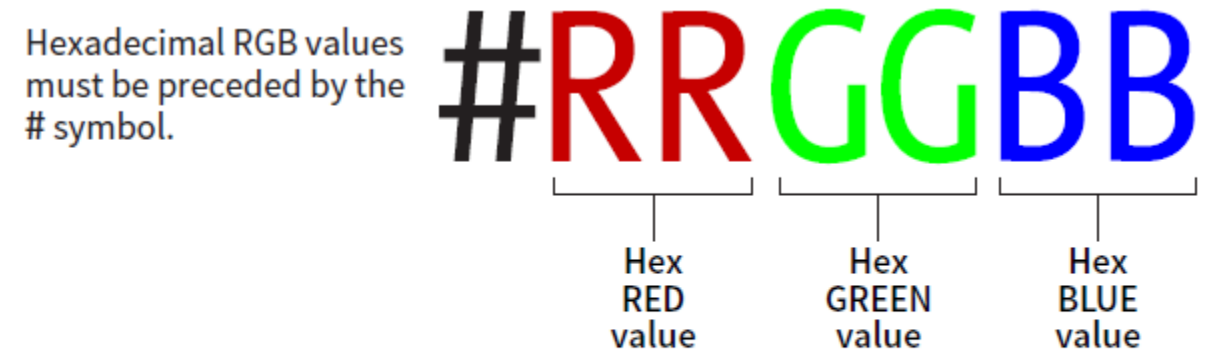


FIGURE 13-5. Hexadecimal RGB values are made up of three two-digit numbers, one for red, one for green, and one for blue.

Writing RGB values in style sheets

color: rgb(200, 178, 230);
color: rgb(78%, 70%, 90%);
color: #C8B2E6;
color: #FC0; or color: #936;

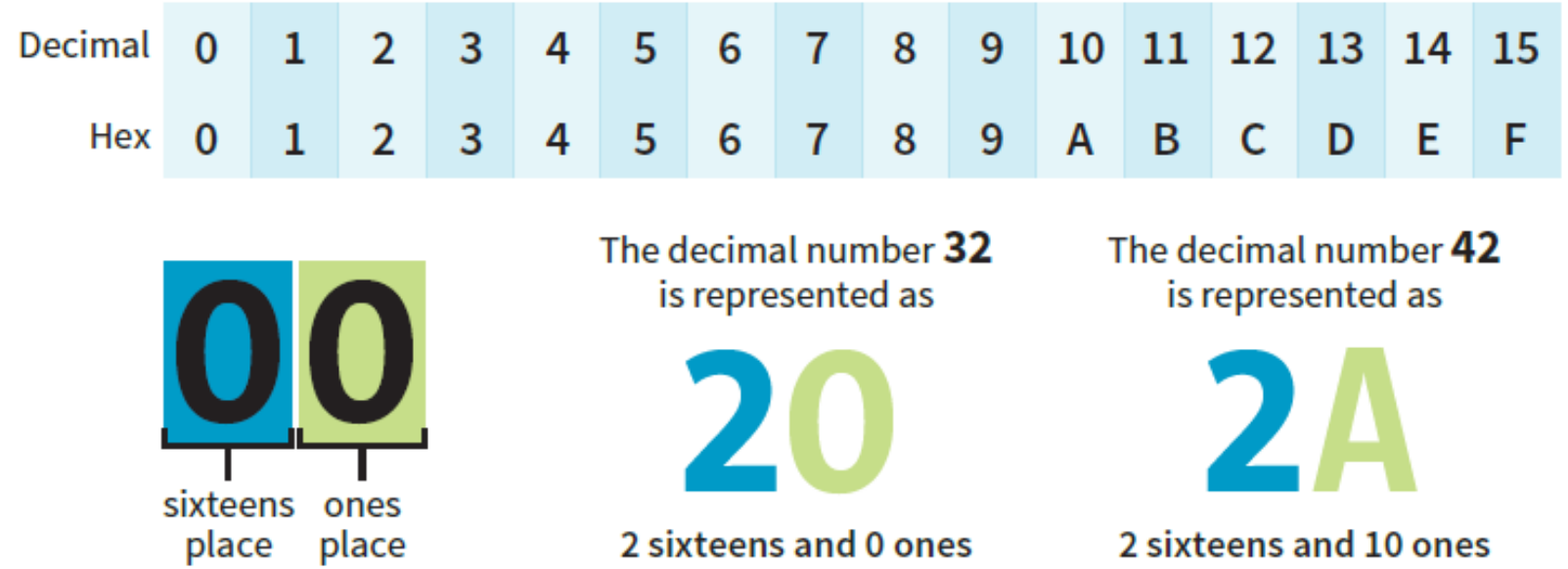
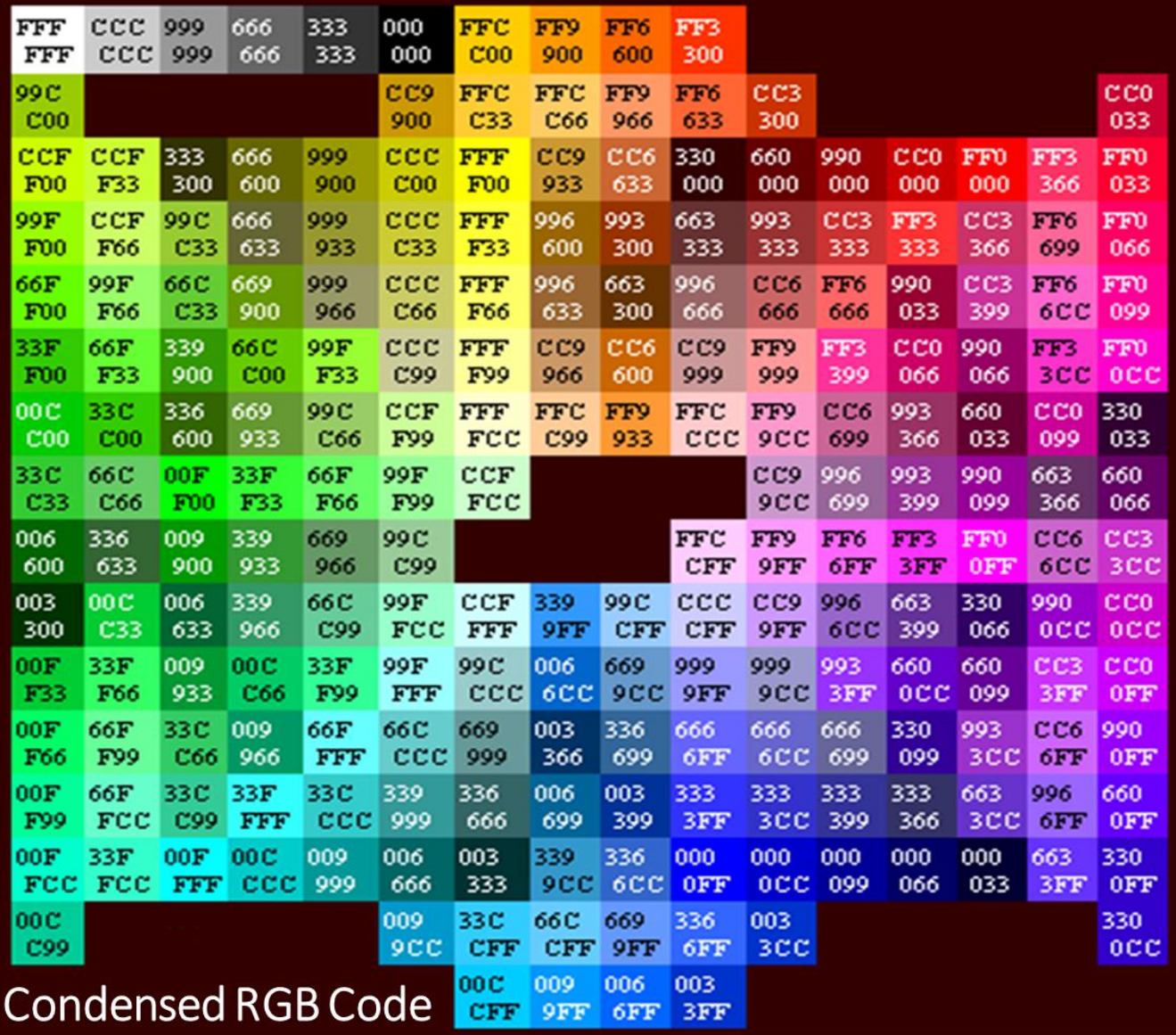
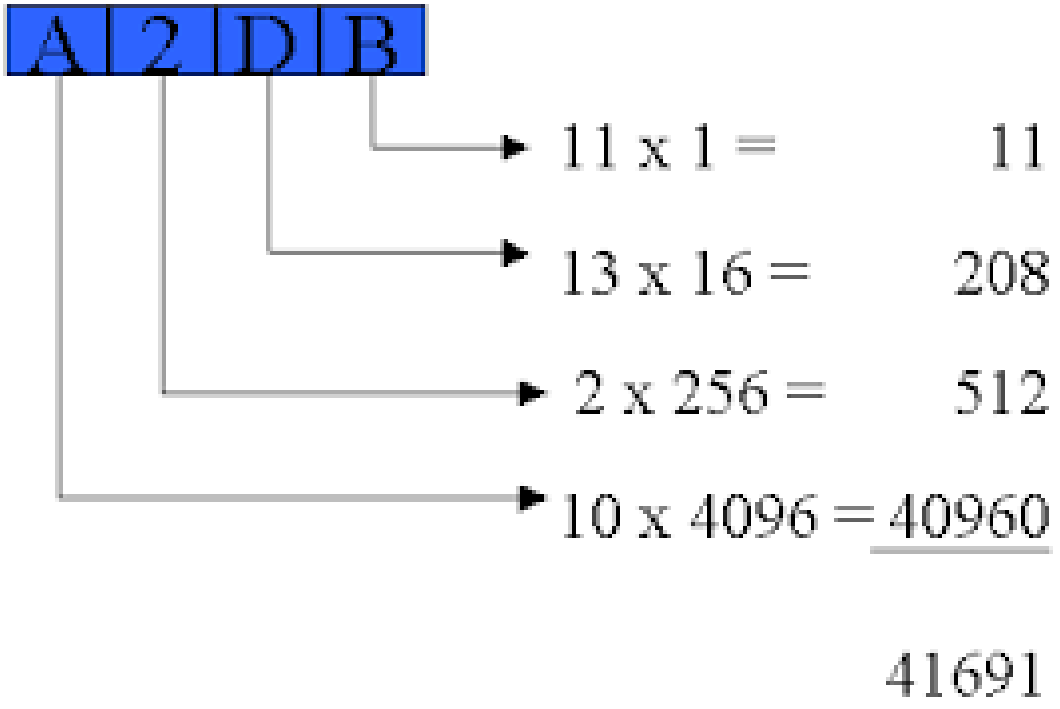


FIGURE 13-6. The hexadecimal numbering system is base-16.

Writing RGB values in style sheets

color: #FC0; or color: #936;





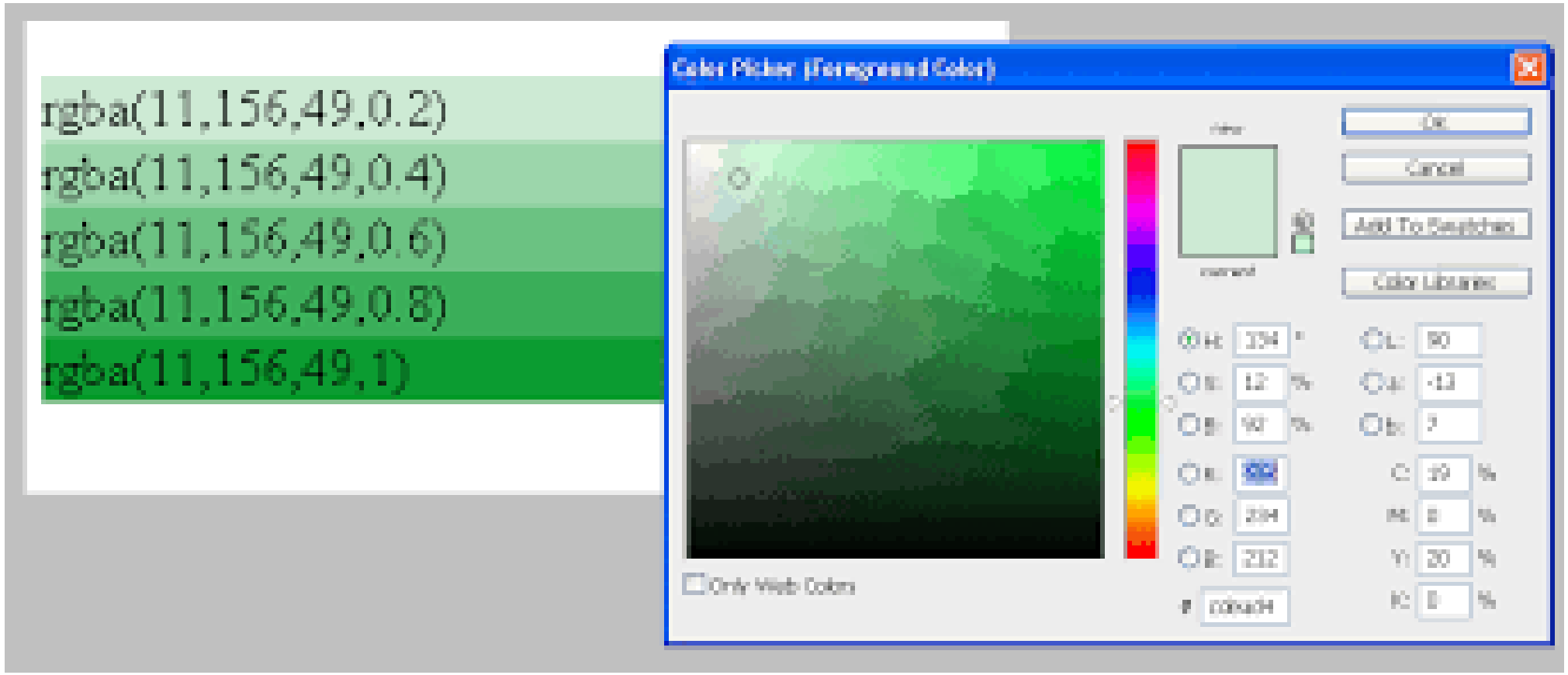
Hexadecimal	Binary	Decimal
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
A	1010	10
B	1011	11
C	1100	12
D	1101	13
E	1110	14
F	1111	15

Hexadecimal Numbers

RGBa Color

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

FIGURE 13-7. Headings with various levels of transparency using RGBa values.

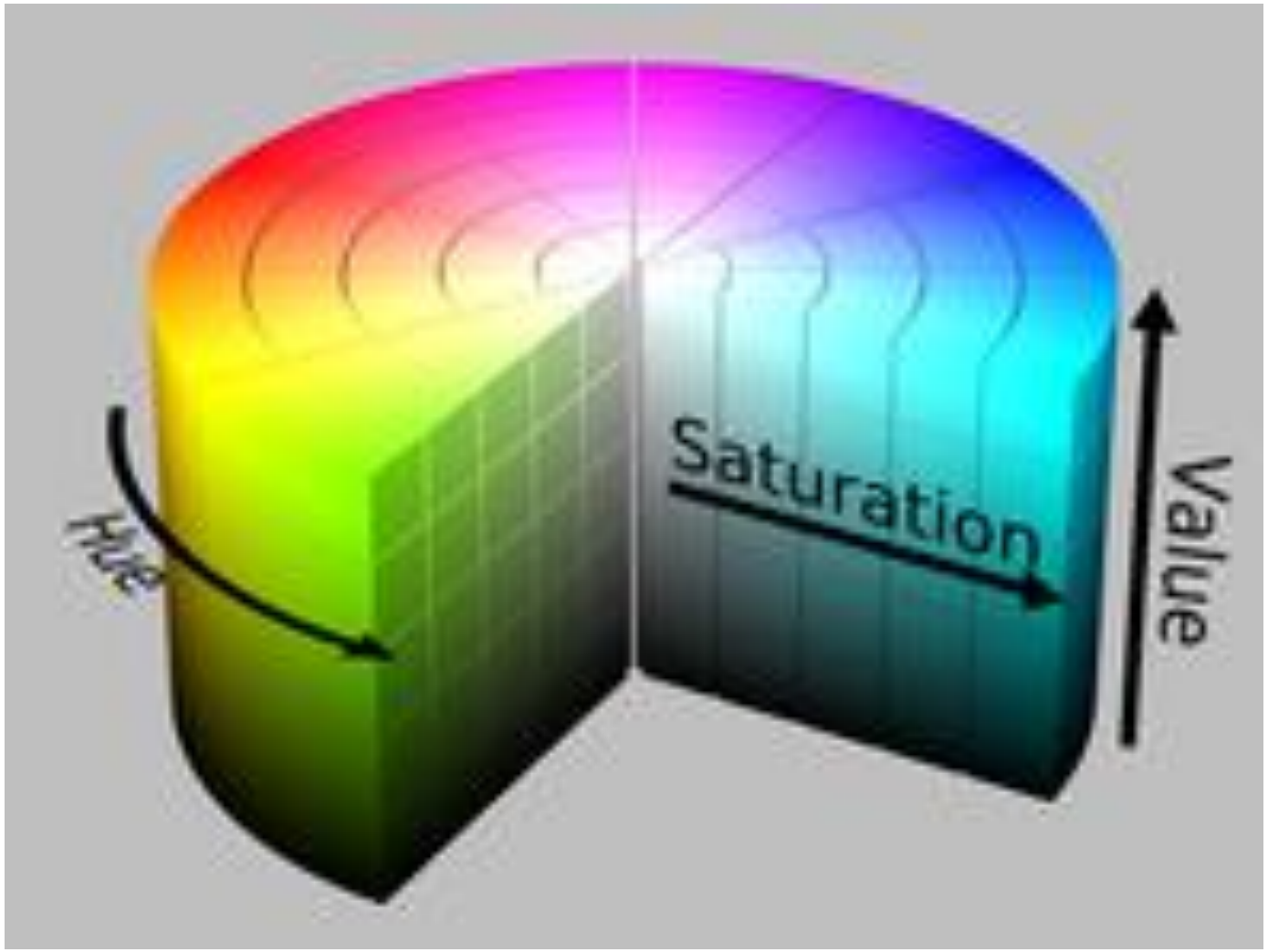


RGBA code (A for alpha channel)

ALPHA CHANNEL IS FOR THE OPAQUE (OPPOSITE OF TRANSPARENCY)

HSL Color

color: hsl(265, 51%, 80%);



HSL Color

color: hsl(265, 51%, 80%);

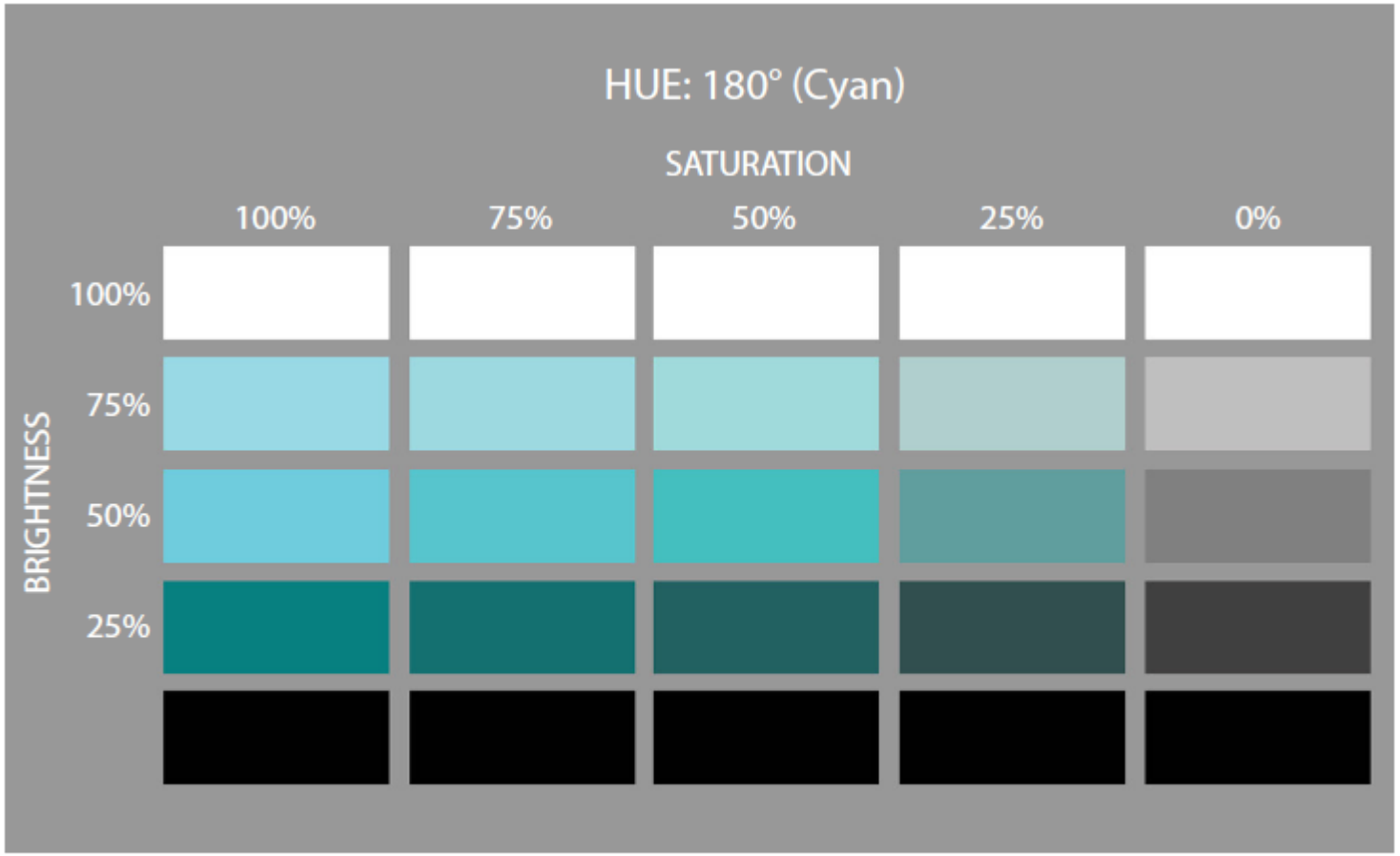


FIGURE 13-8. One hue in the HSL color model, with its associated saturation and lightness values.

Foreground Color

SECTION 10

Foreground, Background, Alpha-channel, and Layers

- HTML does not support Layers. Layers are for other graphics art and image processing.
- **Foreground** means elements of interests.
- **Background** means not changing/moving uninterested things.
- **Layers** mean element or objects of different level of interests. Higher interests are put on top and lower ones are at the bottom (more closer to background). Layers can be raised or lowered.

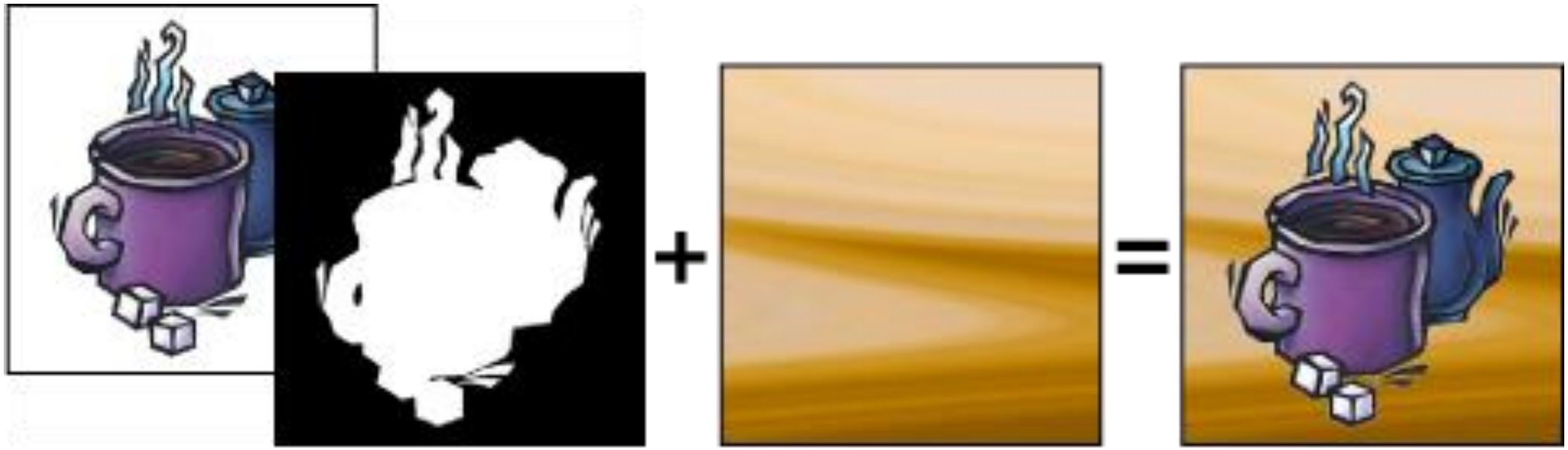
Foreground, Background, Alpha-channel, and Layers

- **Alpha-channel** allow object of higher interests to yield some space for object of lower interests to show.
- The degree of visibility for lower layers is called **transparency**, On the contrary, how solid the higher layer is determined by **opaque** level.

Foreground Color

- The foreground of an element consists of its text and border (if one is specified).
- You specify a foreground color with the color property, as we saw in the last chapter when we rolled it out to give text a little pizzazz.

Foreground, Background, Alpha-channel and Layers

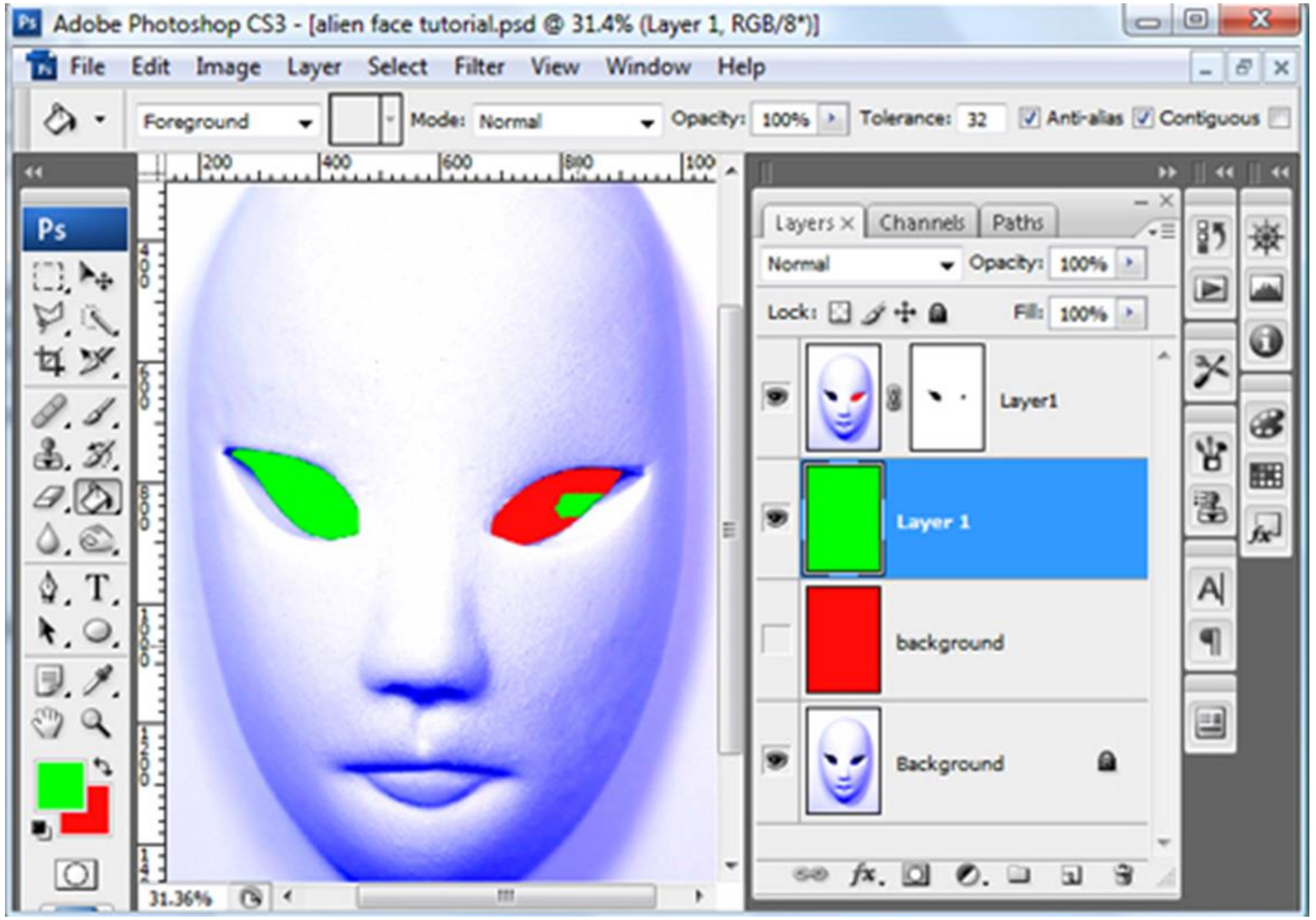


Foreground graphic

Alpha channel

Background graphic

Final composite


Adobe Photoshop CS3 - [alien face tutorial.psd @ 31.4% (Layer 1, RGB/8*)]
File Edit Image Layer Select Filter View Window Help
Foreground
Mode: Normal
Opacity: 100%
Tolerance: 32
Anti-alias
Contiguous
Layers
Channels
Paths
Normal
Opacity: 100%
Lock:
Fill: 100%
Layer1
Layer 1
background
Background
31.36%

color

Values:	*color value (name or numeric)*
Default:	depends on the browser and user's preferences
Applies to:	all elements
Inherits:	yes

color

THE STYLE RULE

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

THE MARKUP

```
<blockquote>
In the latitude of central New England, cabbages are not secure ...
</blockquote>
```

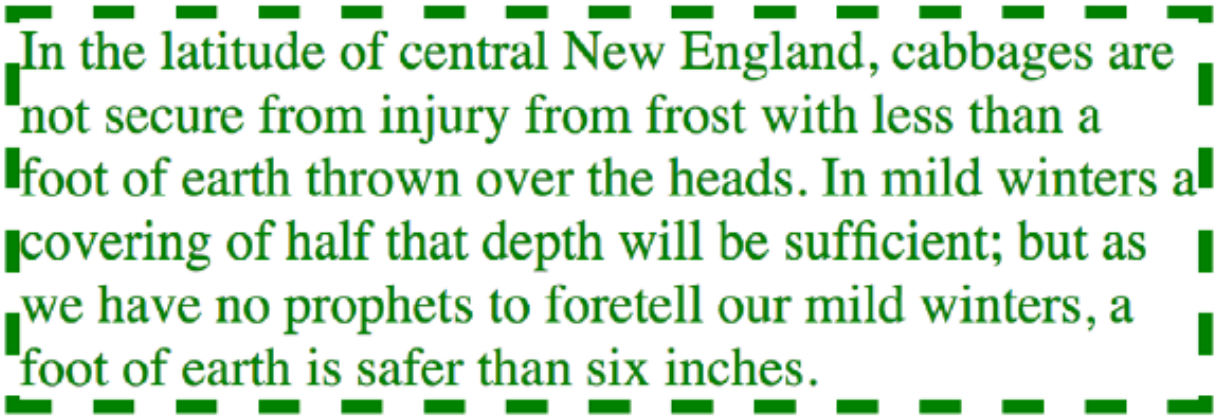


FIGURE 13-9. Applying a color to the foreground of an element.

color

Background Color

SECTION 11

background-color

Values:	*color value (name or numeric)* | transparent
Default:	transparent
Applies to:	all elements
Inherits:	no

Background color

```
blockquote {
  border: 4px dashed;
  color: green;
  background-color: #c6de89;
}
```

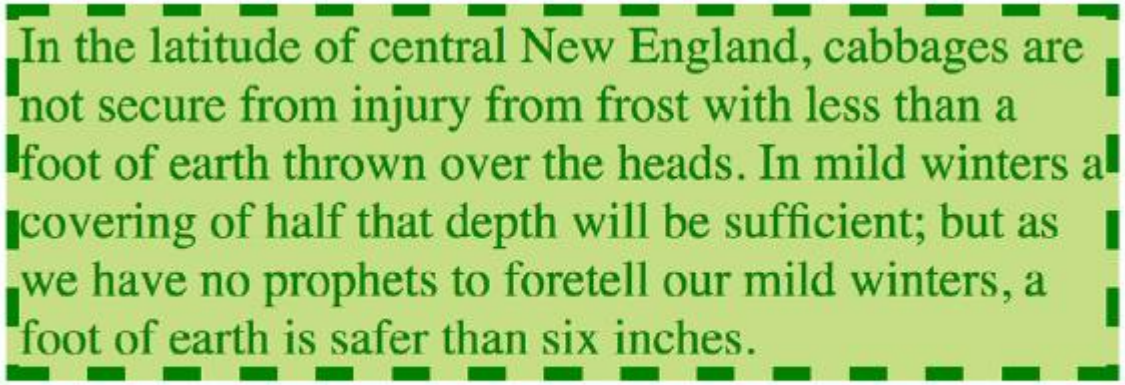


FIGURE 13-10. Adding a light green background color to the sample blockquote.

Background color

THE STYLE RULE

```
.glossary {
  color: #0378a9;  /* blue */
  background-color: yellow;
}
```

THE MARKUP

```
<p>Every variety of cabbage had their origin in the wild cabbage of
Europe (<dfn class="glossary"><i>Brassica oleracea</i></dfn>)</p>
```

Every variety of cabbage had their origin in the wild cabbage of Europe (*Brassica oleracea*)

FIGURE 13-11. Applying the background-color property to an inline element.

Background color

To color the background of the whole page, apply the background-color property to the body element.

background-clip

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

Background-clip

```
blockquote {
  padding: 1em; border: 4px dashed; color: green; background-color: #C6DE89;}
}
```

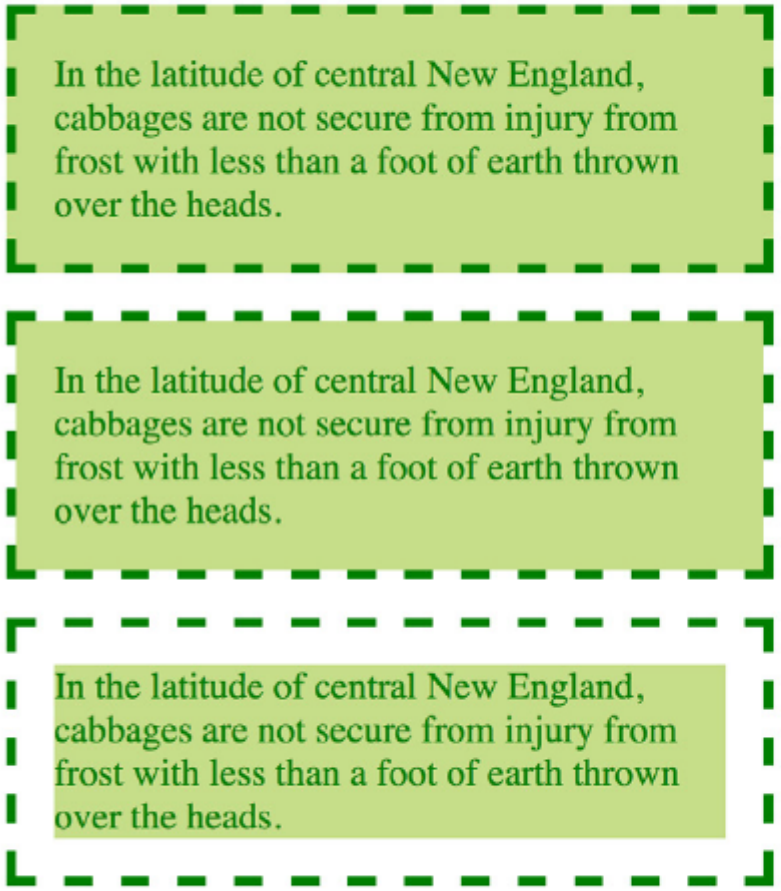


FIGURE 13-12. The **background-clip** property.

Background-clip

opacity

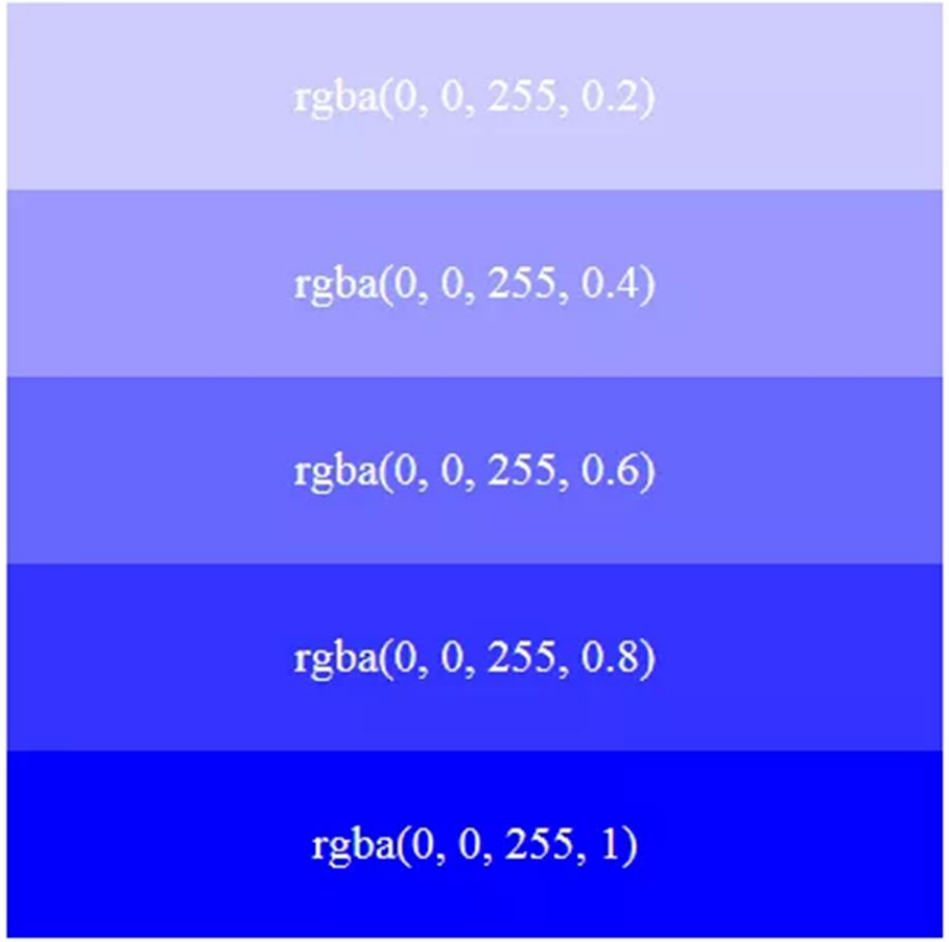
Values:	*number* (0 to 1)
Default:	1
Applies to:	all elements
Inherits:	no

Playing with Opacity

```
h1 {color: gold; background: white; opacity: .25;}
h1 {color: gold; background: white; opacity: .5;}
h1 {color: gold; background: white; opacity: 1;}
```

FIGURE 13-13. Setting the opacity on an element affects both the foreground and background colors.

Playing with Opacity



Playing with Opacity

Advanced Selector

SECTION 12

Selector Review

Here is a quick summary of the selector types we've covered already ("E" stands for "Element"):

Element type selector

E {property: value;}

Grouped selectors

E1, E2, E3 {property: value;}

Descendant selector

E1 E2 {property: value;}

Child selector

E1 > E2 {property: value;}

Next-sibling selector

E1 + E2 {property: value;}

Subsequent-sibling selector

E1 ~ E2 {property: value;}

ID selector

E#id {property: value;}

#id {property: value;}

Class selector

E.class {property: value;}

.class {property: value;}

Universal selector

* {property: value;}

More on Selector

Pseudo Class Selector:

link pseudo class selector, action pseudo class selector

Pseudo Element Selector:

first letter selector, first line selector, before, end selector

Attribute Selector:

Element[Attribute] and many others.

Pseudo Class

- Have you ever noticed that a link is often one color when you click it and another color when you go back to that page? That's because, behind the scenes***, your browser is keeping track of which links have been clicked*** (or "visited," to use the lingo).
- The browser keeps track of other states too, such as whether the user's cursor is over an element (hover state), whether an element is the first of its type, whether it's the first or last child of its parent, and whether a form element has been checked or disabled, just to name a few.

Pseudo Class

- In CSS, you can apply styles to elements in these states by using a special kind of selector called a **pseudo-class** selector.
- It’s an odd name, but you can think of it as though elements in **a certain state** belong to the same class. However, the class name isn’t in the markup—it’s something the browser just keeps track of.
- So it’s kinda like a class…it’s a pseudo-class.

Pseudo Class

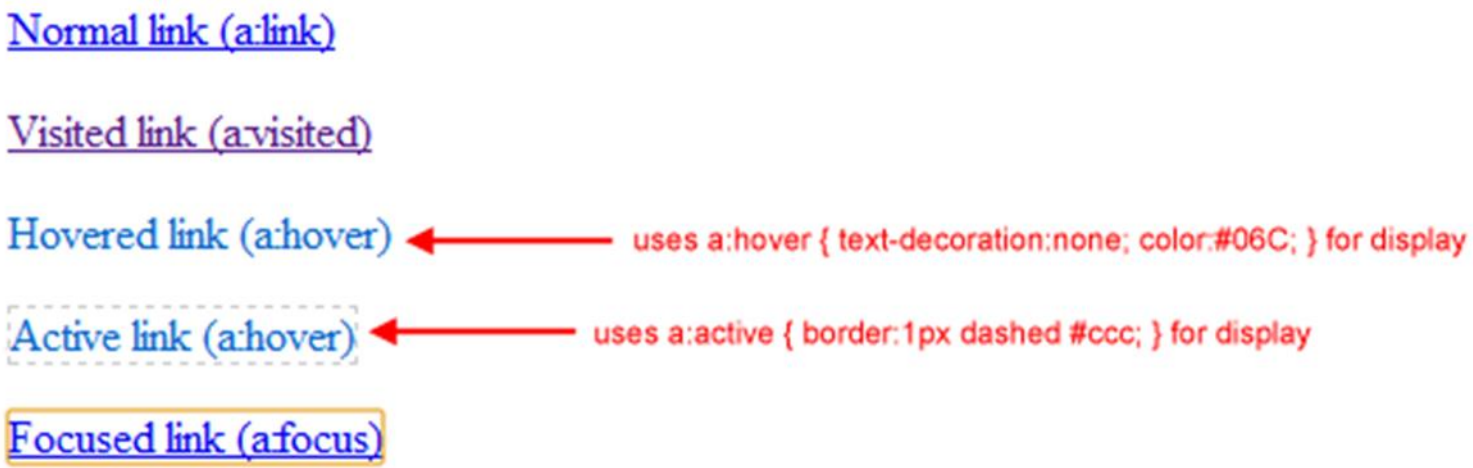
- Pseudo-class selectors are indicated by the colon (**:**) character.
- They typically go immediately after an element name—for example, **li:first-child**.

Link Pseudo-Classes

:link Apply the style setting to unvisisted links

:visited Apply the style setting to visited links

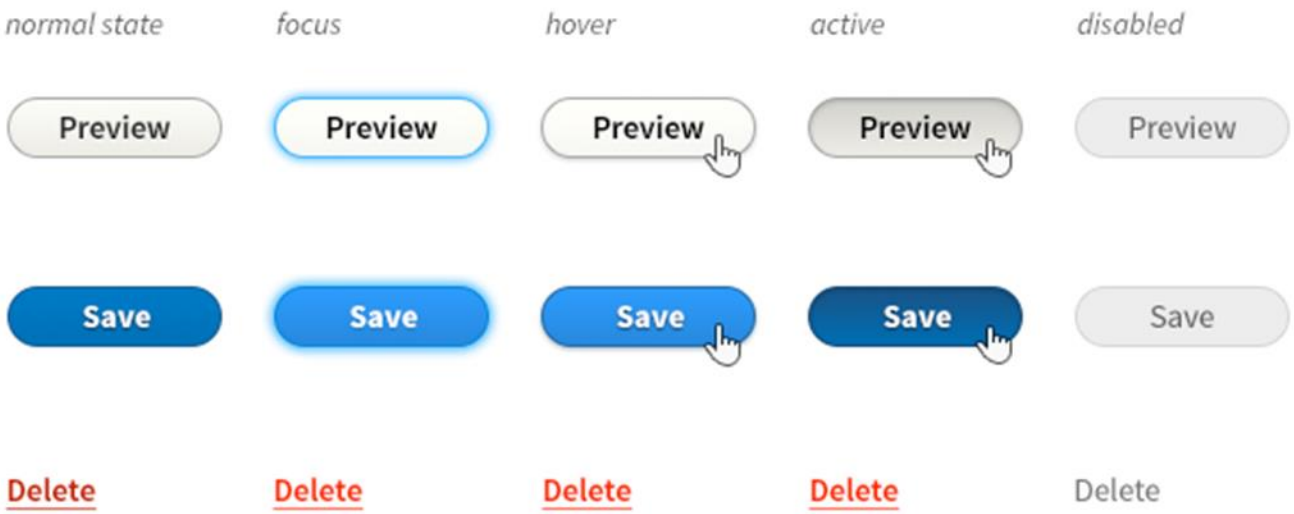
```
a:link { color: maroon; }
a:visited { color:gray; }
```



Link Pseudo-Classes

Action Pseudo-classes

:focus	Applies when the element is selected and ready for input
:hover	Applies when the mouse pointer is over the element
:active	applies when the element (such as link or buttons) is in the process of being clicked or tapped



Link Pseudo-Classes

More CSS3 Pseudo-Classes

Structural pseudo-classes
These allow selection based on where the element is in the structure of the document (the document tree):

- :root
- :empty
- :first-child
- :last-child
- :only-child
- :first-of-type
- :last-of-type
- :only-of-type
- :nth-child()
- :nth-last-child()
- :nth-of-type()
- :nth-last-of-type()

Input pseudo-classes
These selectors apply to states that are typical for form inputs:

- :enabled
- :disabled
- :checked

Location pseudo-classes (in addition to :link and :visited)

- :target (fragment identifier)

Linguistic pseudo-class

- :lang()

Logical pseudo-class

- :not()

Pseudo Element Selector

- There are also four pseudo-elements that act as though they are inserting fictional elements into the document structure for styling.
- In CSS3, pseudo-elements are indicated by a double colon (::) symbol to differentiate them from pseudo-classes.
- However, all browsers support the single-colon syntax (:) as they were defined in CSS2, so ***many developers stick with that to ensure backward compatibility with older browsers***.

::first-line

This selector applies a style rule to the first line of the specified element. The only properties you can apply, however, are as follows:

```
color
font properties
background properties
word-spacing
letter-spacing
text-decoration
vertical-align
text-transform
line-height
```

First Line

::first-line

::first-letter

This applies a style rule to the first letter of the specified element. The properties you can apply are limited to the following:

```
color
font properties
background properties
letter-spacing
word-spacing
text-decoration
text-transform
vertical-align (if float is none)
padding properties
margin properties
border properties
line-height
float
```

First Letter

::first-letter

```
p::first-line { letter-spacing: 9px; }
p::first-letter { font-size: 300%; color: orange; }
```

::first-line

I n s o m e o f t h e b e s t c a b b a g e -growing sections of the country, until within a comparatively few years it was the very general belief that cabbage would not do well on upland. Accordingly the cabbage patch would be found on the lowest tillage land of the farm.

::first-letter

In some of the best cabbage-growing sections of the country, until within a comparatively few years it was the very general belief that cabbage would not do well on upland. Accordingly the cabbage patch would be found on the lowest tillage land of the farm.

FIGURE 13-15. Examples of **::first-line** and **::first-letter** pseudo-element selectors.

First Letter and Line

::first-line
::first-letter

Generated Content with ::before and ::after

- You've seen how browsers add bullets and numbers to lists automatically, even though they are not actually in the HTML source. That is an example of generated content, content that browsers insert on the fly. It is possible to tell browsers to generate content before or after any element you like by using the **::before** and **::after** pseudo-elements (see Note).
- Generated content could be used to add icons before list items, to display URLs next to links when web documents get printed out, to add language appropriate quotation marks around a quote, and much more. Here's a simple example that inserts an image by using the **url()** function before the paragraph and "Thank you." at the end of the paragraph. Compare the markup to what you see rendered in the browser (FIGURE 13-16).

THE STYLES:

```
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>
```

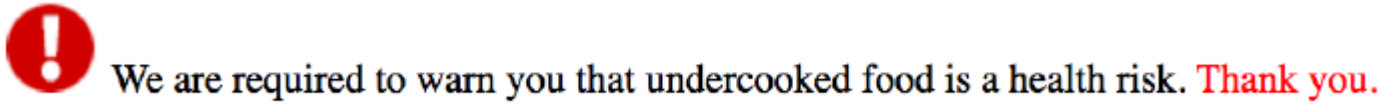


FIGURE 13-16. Generated content added with the `::before` and `::after` pseudo-selectors.

Before and After a class

::before
::after

Attribute Selector

- Attribute selectors target elements based on attribute names or values, which provides a lot of flexibility for selecting elements without needing to add a lot of **class** or **id** markup. The CSS3 attribute selectors are listed here:

element[*attribute***]**

The simple attribute selector targets elements with a particular attribute regardless of its value. The following example selects any image that has a **title** attribute.

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

element[*attribute***="***exact value***"]**

The exact attribute value selector selects elements with a specific value for the attribute. This selector matches images with exactly the **title** value “first grade”.

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

Attribute Selector

element[*attribute*~="*value*"]

The partial attribute value selector (indicated with a tilde, **~**) allows you to specify one part of an attribute value. The following example looks for the word “grade” in the title, so images with the **title** value “first grade” and “second grade” would be selected.

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

element[*attribute*|="*value*"]

The hyphen-separated attribute value selector (indicated with a bar, **|**) targets hyphen-separated values. This selector matches any link that points to a document written in a variation on the English language (**en**), whether the attribute value is **en-us** (American English), **en-in** (Indian English), **en-au-tas** (Australian English), and so on.

```
[hreflang|="en"] {border: 3px solid;}
```

Attribute Selector

element[*attribute*^="*first part of the value*"**]**

The beginning substring attribute value selector (indicated with a carat, **^**) matches elements whose specified attribute values *start* in the string of characters in the selector. This example applies the style only to images that are found in the */images/icons* directory.

```
img[src^="/images/icons"] {border: 3px solid;}
```

element[*attribute*$="*last part of the value*"**]**

The ending substring attribute value selector (indicated with a dollar sign, **$**) matches elements whose specified attribute values *end* in the string of characters in the selector. In this example, you can apply a style to just the **a** elements that link to PDF files.

```
a[href$=".pdf"] {border-bottom: 3px solid;}
```

Attribute Selector

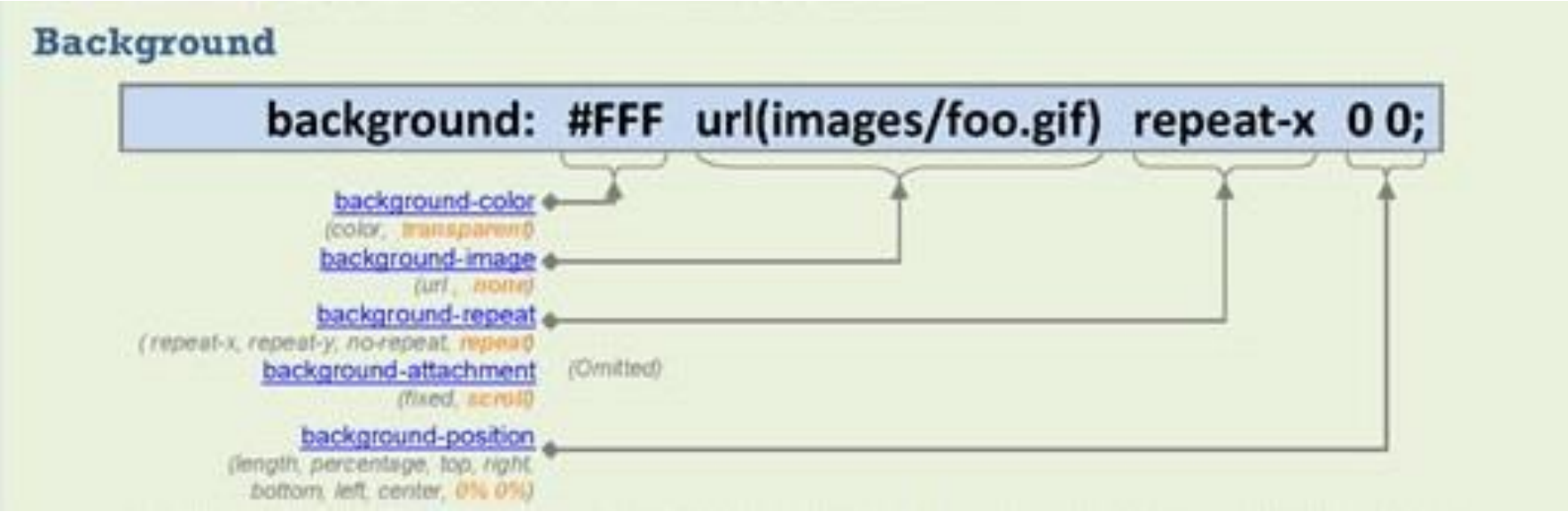
element[*attribute**="***any part of the value***"]**

The arbitrary substring attribute value selector (indicated with an asterisk, *****) looks for the provided text string in any part of the attribute value specified. This rule selects any image that contains the word “February” somewhere in its **title**.

```
img[title*="February"] {border: 3px solid;}
```

Background Images

SECTION 13



CSS: background shorthand

Background Properties

background-color: color setting when there is no image used for background

background-image: image or images used for background purpose

background-repeat: how to repeat the background image

background-attachment: how the background works with foreground when the viewport scrolls

background-position: where should the background image be placed

background-clip: this specify how far the background image should extend.

background-size: this property allows designers to size the background image inside the element.

background-origin: this property determines how the background-position is calculated, or in other words, where to start counting positioning measurements.

Adding a Background Image

- The **background-image** property adds a background image to any element. Its primary job is to provide the location of the image file.

background-image

Values:	url(*location of image*) | none
Default:	none
Applies to:	all elements
Inherits:	no

- The value of **background-image** is a sort of URL holder that contains the location of the image

Adding a Background Image

- Background-image: url(location of image file) | none | inherit

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

```
body {
   background-image: url(star.png);
}
blockquote {
   background-image: url(dot.png);
   padding: 2em;
   border: 4px dashed;
}
```

Adding a Background Image

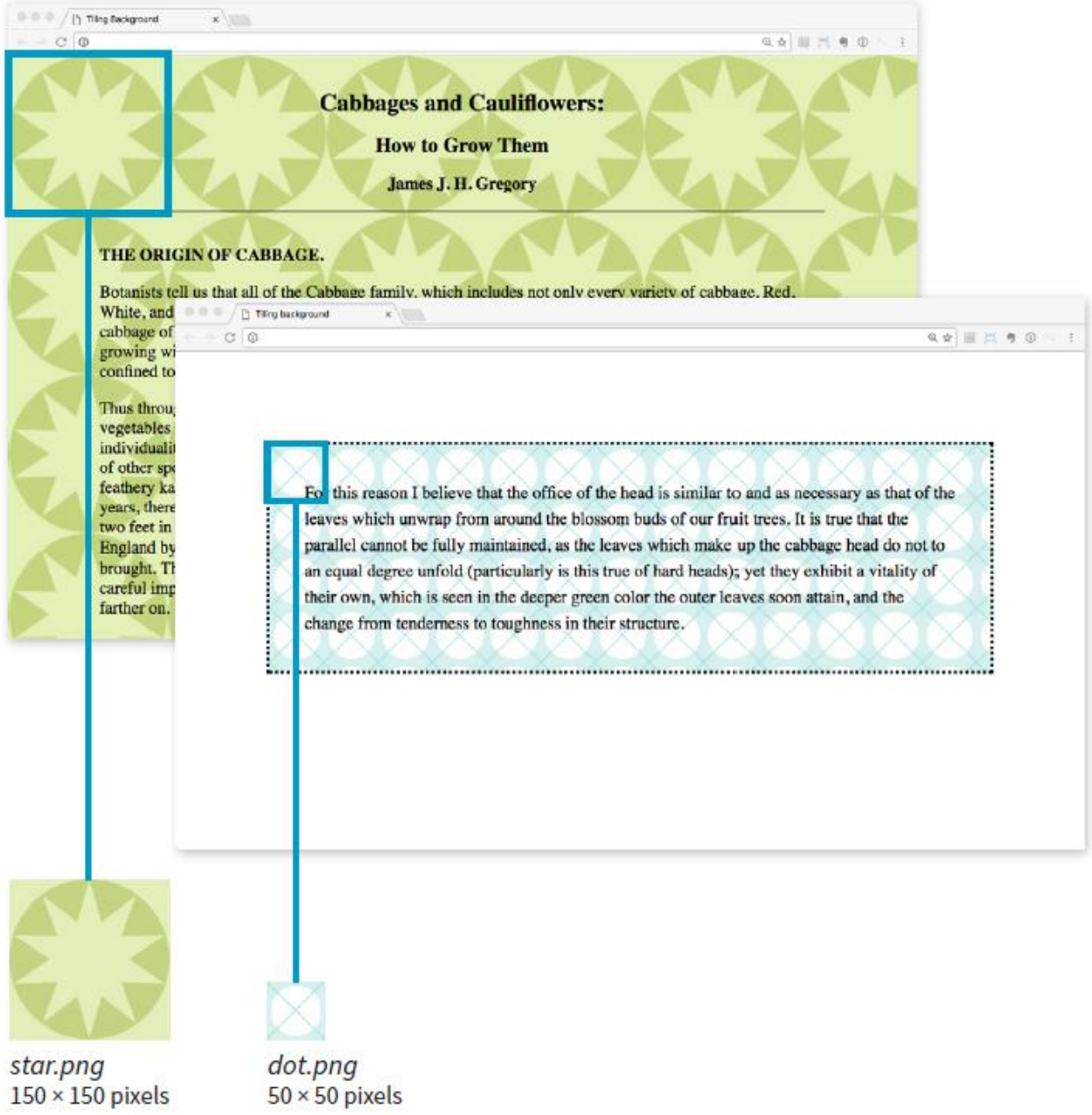


FIGURE 13-18. Tiling background images added with the **background-image** property.

Adding a Background Image

background-repeat

Values: repeat | no-repeat | repeat-x | repeat-y | space | round

Default: repeat

Applies to: all elements

Inherits: no

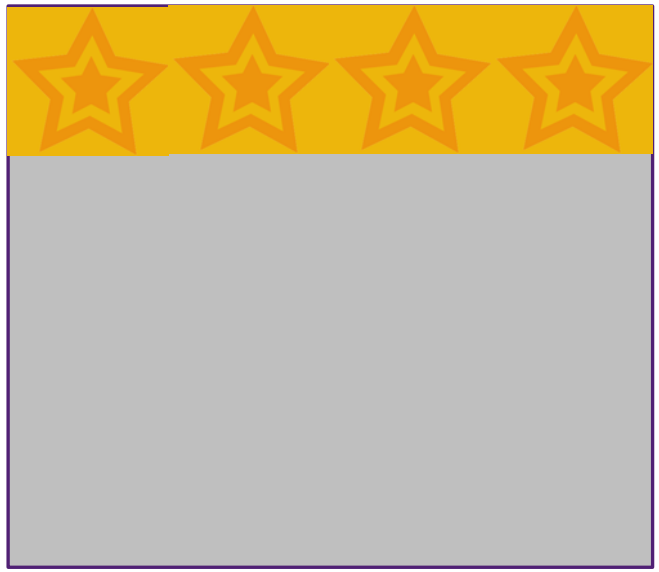
Background Repeating

```
body {
    background-image: url(star.png);
    background-repeat: no-repeat;
}
```

Background Repeating

no-repeat

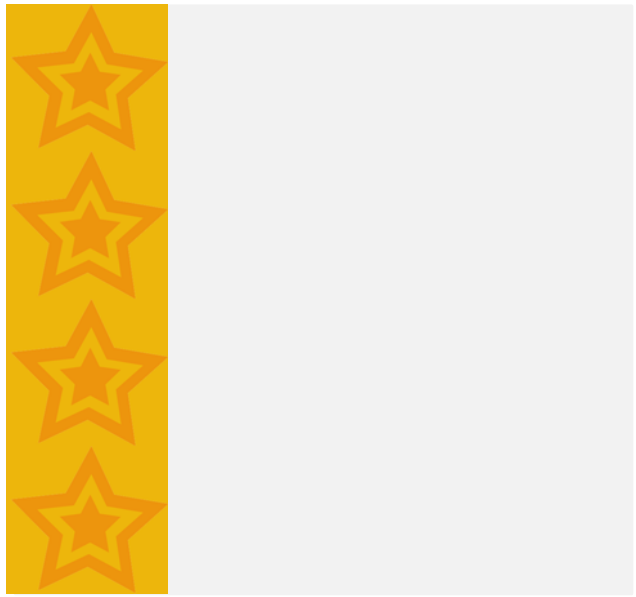
```
body {
    background-image: url(star.png);
    background-repeat: repeat-x;
}
```



Background Repeating

repeat-x

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



Background Repeating

repeat-y

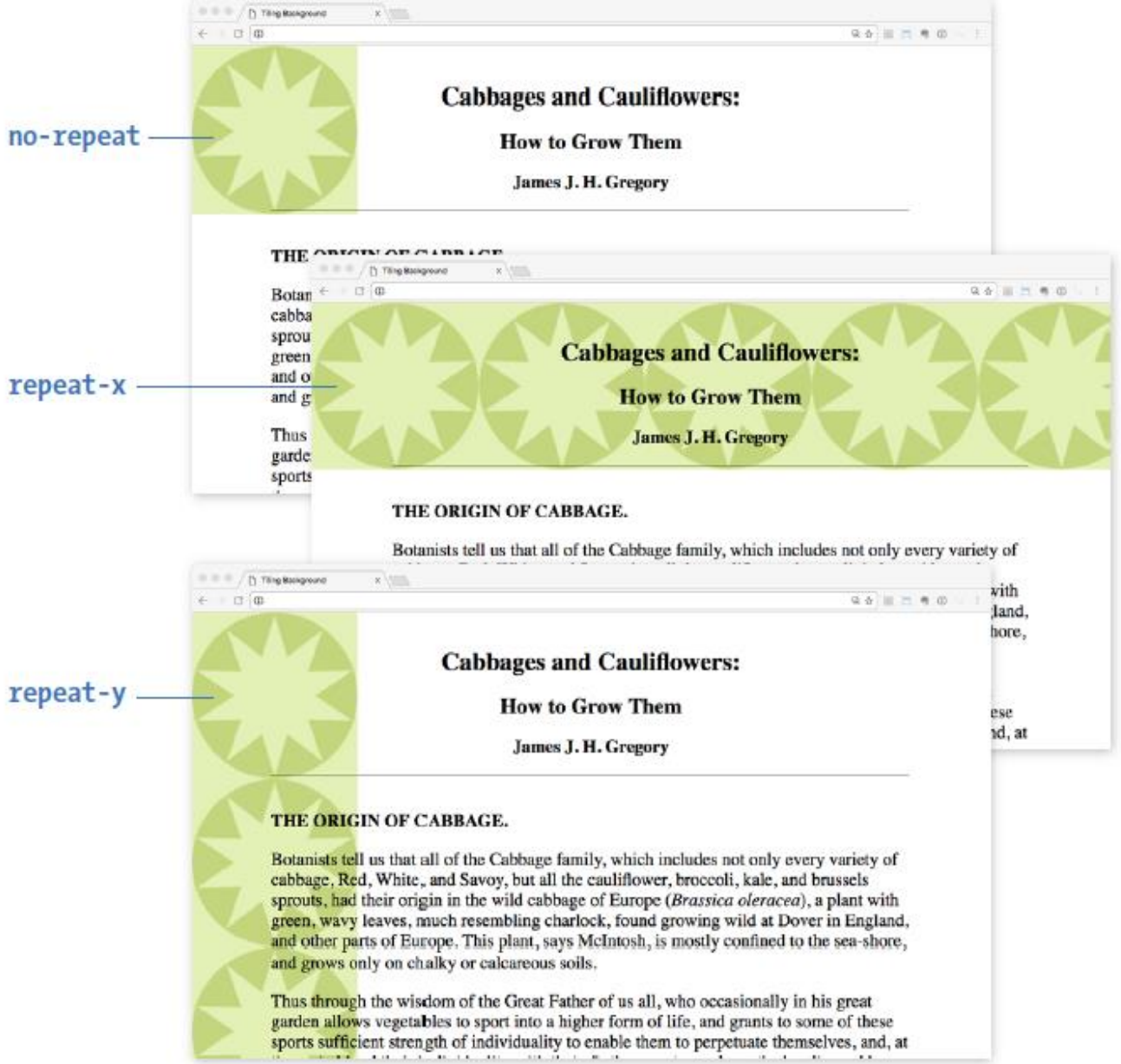


FIGURE 13-20. Turning off automatic tiling with **no-repeat** (top), applying horizontal-axis tiling with **repeat-x** (middle), and applying vertical-axis tiling with **repeat-y** (bottom).

Background Repeating

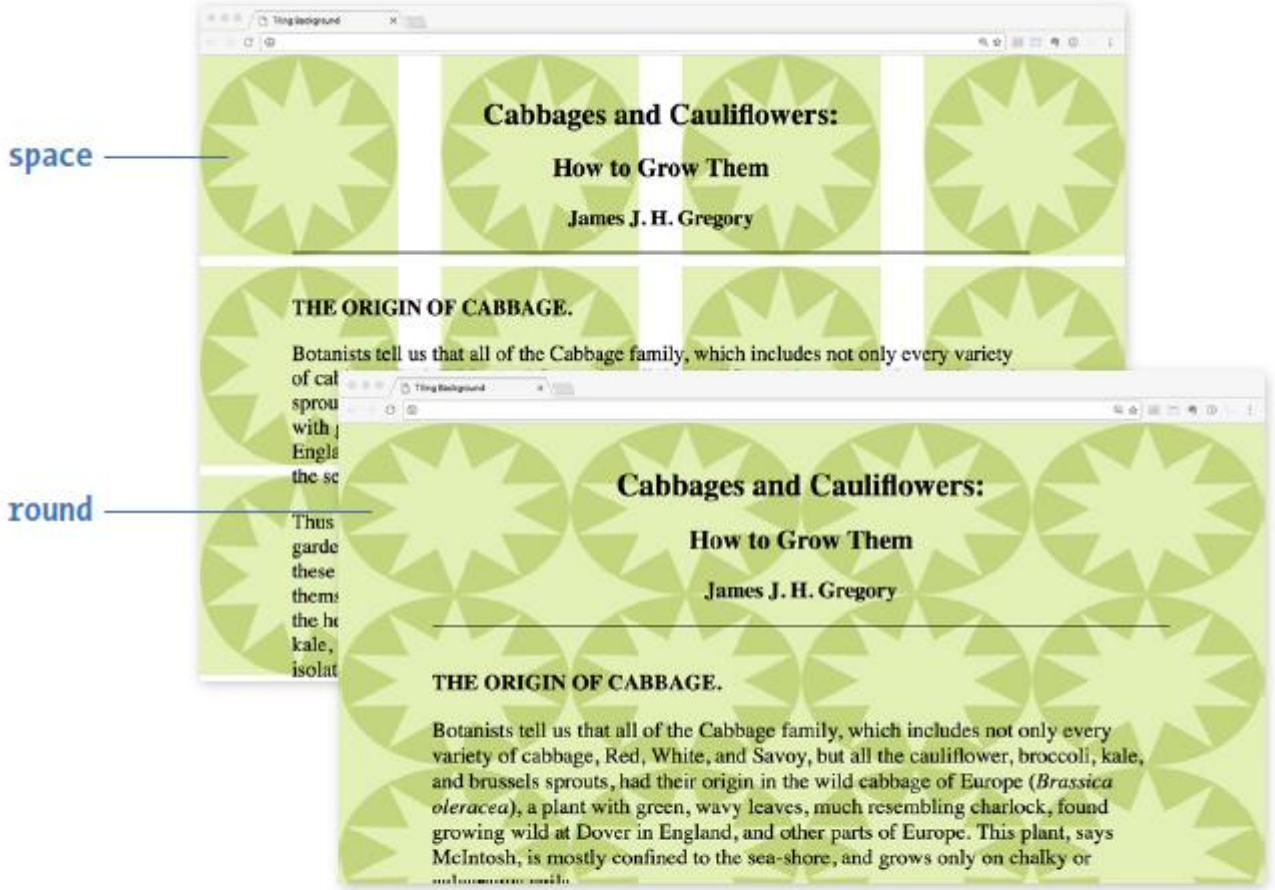


FIGURE 13-21. Examples of **space** and **round** keywords for **background-repeat**. The "space" example would be less clunky if the background color matched the image, but I've left it white to better demonstrate how the **space** value works.

Background Repeating

Background Position

The **background-position** property specifies the position of the origin image in the background. You can think of the origin image as the first image that is placed in the background from which tiling images extend. Here is the property and its various values.

background-position

Values: *length measurement* | *percentage* | left | center | right | top | bottom
Default: 0% 0% (same as left top)
Applies to: all elements
Inherits: no

- To position the origin image, provide horizontal and vertical values that describe where to place it. There are a variety of ways to do it.

background-position: left bottom;

background-position: right center;

background-position: 200px 50px;

background-position: 15% 100%;

background-position

Background-position: length | percentage | left | center | right | top | bottom | inherit

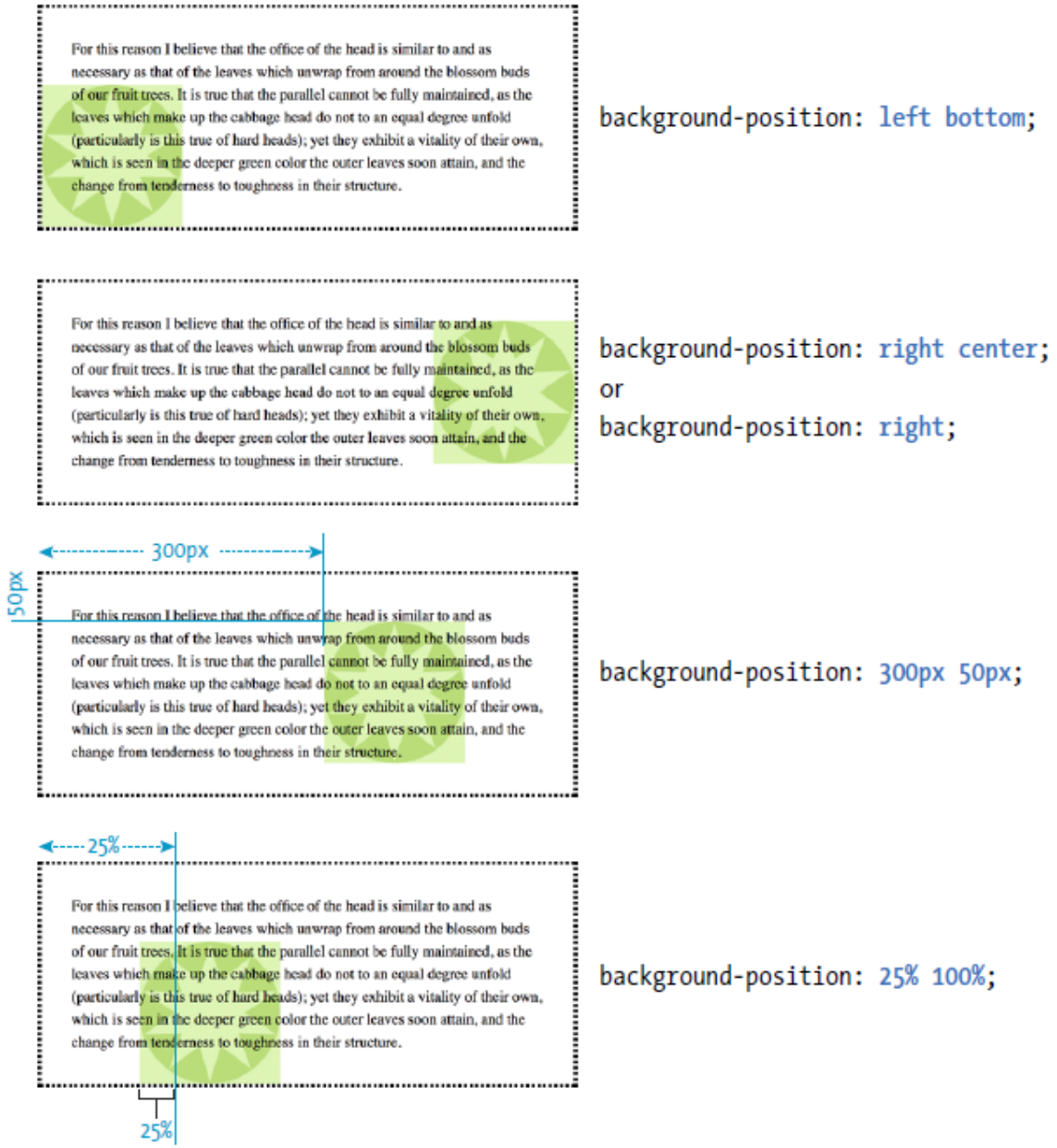


FIGURE 13-23. Positioning a non-repeating background image. If these background images were allowed to repeat, they would extend left and right and/or up and down from the initial positions.

background-position

Background-position: length | percentage | left | center | right | top | bottom | inherit

background-origin

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

- This property defines the boundaries of the background positioning area in the same way background-clip defined the background painting area. You can set the boundaries to the border-box (so the origin image is placed under the outer edge of the border, padding-box (outer edge of the padding, just inside the border), or content-box (the actual content area of the element).
- These terms will become more meaningful once you get more familiar with the box model in the next chapter.

Background Position Origin

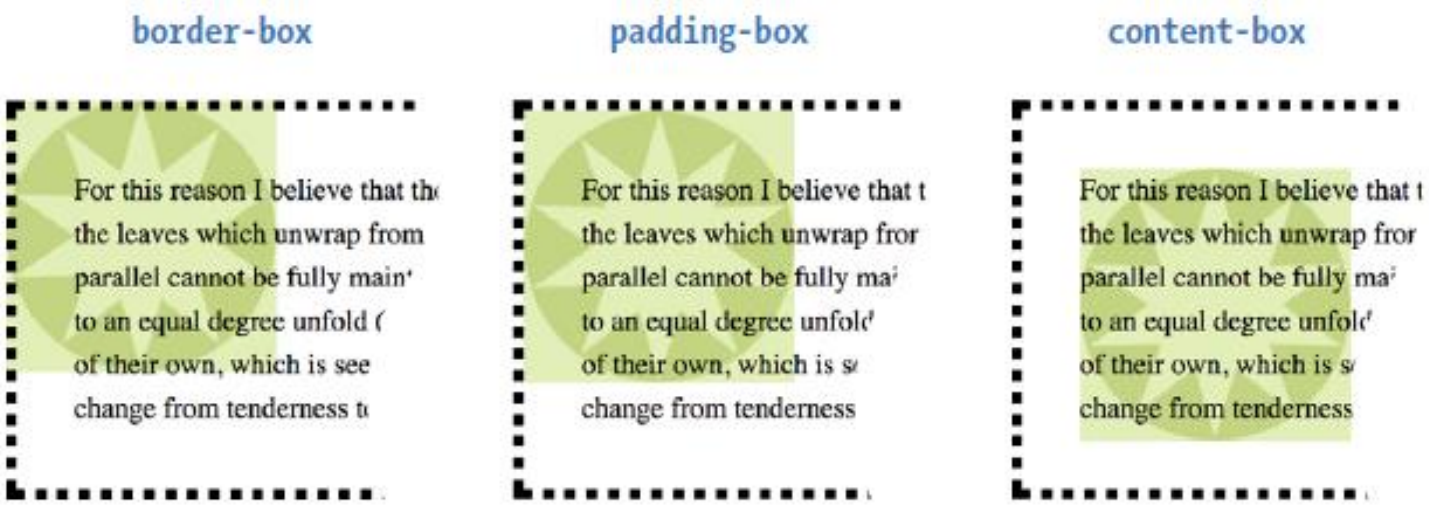


FIGURE 13-24. Examples of **background-origin** keywords.

Background Position Origin

background-attachment

Values:	scroll | fixed | local
Default:	scroll
Applies to:	all elements
Inherits:	no

- With the background-attachment property, you have the choice of whether the background image scrolls with the content or stays in a fixed position. When an image is fixed, it stays in the same position relative to the viewport of the browser

```
body {
    background-image:
    url(images/bigstar.gif);
    background-repeat: no-repeat;
    background-position: center 300px;
    background-attachment: fixed;
}
```

Background Attachment

Background Attachment

Cabbages and Cauliflowers:

How to Grow Them

James J. H. Gregory

THE ORIGIN OF CABBAGE.

Botanists tell us that all of the Cabbage family, which includes not only every variety of cabbage, Red, White, and Savoy, but all the cauliflower, broccoli, kale, and brussels sprouts, had their origin in the wild cabbage of Europe (*Brassica oleracea*), a plant with green, wavy leaves, much resembling charlock, found growing wild at Dover in England and other parts of Europe. This plant, says McIntosh, is mostly confined to the sea-shore, and grows only on chalky or calcareous soils.

Thus through the wisdom of the Great Father of us all, who occasionally in his great garden allows vegetables to sport into a higher form of life, and grants to some of these sports sufficient strength of individuality to enable them to perpetuate themselves, and, at times, to blend their individuality with that of other sports, we have the heading cabbage in its numerous varieties, the crumpy cauliflower, the feathery kale, the curled savoy. On my own grounds from a strain of seed that had been grown isolated for years, there recently came a plant that in its structure closely resembled Brussels Sprouts, growing about two feet in height,

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

and brussels sprouts, had their origin in the wild cabbage of Europe (*Brassica oleracea*), a plant with green, wavy leaves, much resembling charlock, found growing wild at Dover in England and other parts of Europe. This plant, says McIntosh, is mostly confined to the sea-shore, and grows only on chalky or calcareous soils.

Thus through the wisdom of the Great Father of us all, who occasionally in his great garden allows vegetables to sport into a higher form of life, and grants to some of these sports sufficient strength of individuality to enable them to perpetuate themselves, and, at times, to blend their individuality with that of other sports, we have the heading cabbage in its numerous varieties, the crumpy cauliflower, the feathery kale, the curled savoy. On my own grounds from a strain of seed that had been grown isolated for years, there recently came a plant that in its structure closely resembled Brussels Sprouts, growing about two feet in height, with a small head under each leaf. The cultivated cabbage was first introduced into England by the Romans, and from there nearly all the kinds cultivated in this country were originally brought. Those which we consider as peculiarly American varieties, have only been made so by years of careful improvement on the original imported sorts. The characteristics of these varieties will be given further on.

WHAT A CABBAGE IS.

If we cut vertically through the middle of the head, we shall find it made up of successive layers of leaves, which grow smaller and smaller, almost ad infinitum. Now, if we take a fruit bud from an apple-tree and make a similar section of it, we shall find the same structure. If we observe the development of the two, as spring advances, we shall find another similarity (the looser the head the closer will be

`background-attachment: scroll;`

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

Now, if we take a fruit bud from an apple-tree and make a similar section of it, we shall find the same structure. If we observe the development of the two, as spring advances, we shall find another similarity (the looser the head the closer will be the resemblance),—the outer leaves of each will unwrap and unfold, and a flower stem will push out from each. Here we see that a cabbage is a bud, a seed bud (as all fruit buds may be termed, the production of seed being the primary object in nature, the fruit enclosing it playing but a secondary part), the office of the leaves being to cover, protect, and afterwards nourish the young seed shoot. The outer leaves, which surround the head appear to have the same office as the leaves which surround the growing fruit bud, and that office closes with the first year, as does that of the leaves surrounding fruit buds, when each die and dropoff. In my locality the public must have perceived more or less clearly the analogy between the heads of cabbage and the buds of trees, for when they speak of small heads they frequently call them "buds." That the close wrapped leaves which make the cabbage head and surround the seed germ, situated just in the middle of the head at the termination of the stump, are necessary for its protection and nutrition when young, is proved, I think, by the fact that those cabbages, the heads of which are much decayed, when set out for seed, no matter how sound the seed germ may be at the end of the stump, never make so large or healthy a seed shoot as those do the heads of which are sound, as a rule, after pushing a feeble growth, they die.

For this reason I believe that the office of the head is similar to and as necessary as that of the leaves which unwrap from around the blossom buds of our fruit trees. It is true that the parallel cannot be fully maintained, as the leaves which make up the cabbage head do not to an equal degree unfold (particularly is this true of hard heads); yet they exhibit a vitality of their own, which is seen in the deeper green color the outer leaves soon attain, and the change from tenderness to toughness in

`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.

FIGURE 13-26. Preventing the background image from scrolling with the **`background-attachment`** property.

background-size

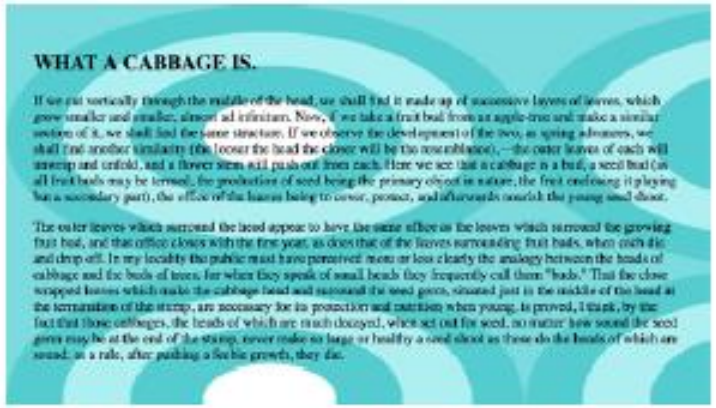
Values:	*length* | *percentage* | auto | cover | contain
Default:	auto
Applies to:	all elements
Inherits:	no

- There are several ways to specify the size of the background image. Perhaps the most straightforward is to specify the dimensions in length units such as pixels or ems. As usual, when two values are provided, the first one is used as the horizontal measurement. If you provide just one value, it is used as the horizontal measurement, and the vertical value is set to auto.

Background Size

```
header {
    background-image: url(images/target.png);
    background-size: 600px 150px;
}
header {
    background-image: url(images/target.png);
    background-size: 50% 10em;
}
```

target.png
300 × 300 pixels



background-size: 600px 300px;

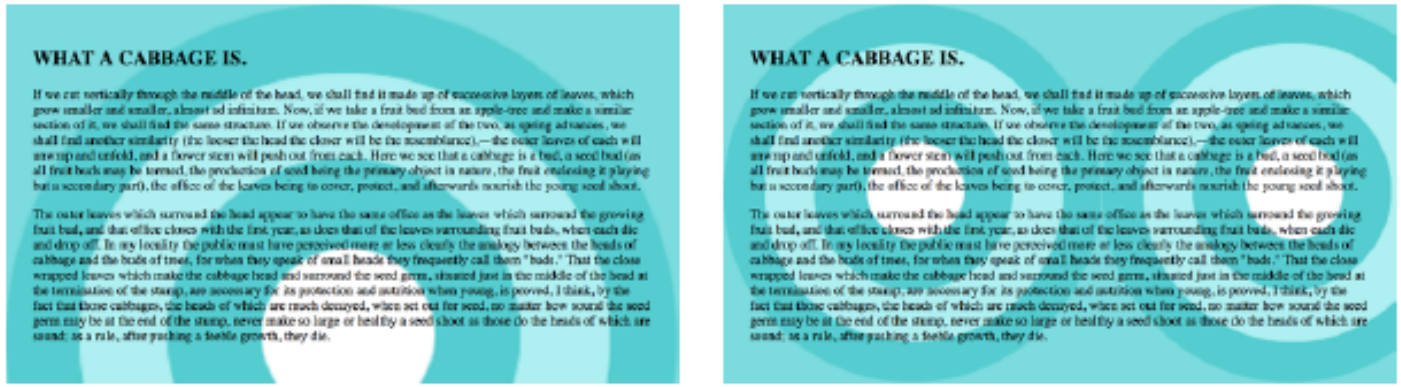
background-size: 50% 10em;

FIGURE 13-27. Resizing a background image with specific length units and percentages.

Background Size

```
div#A {
  background-image: url(target.png);
  background-size: cover; }

div#B {
  background-image: url(target.png);
  background-size: contain; }
```



`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;`

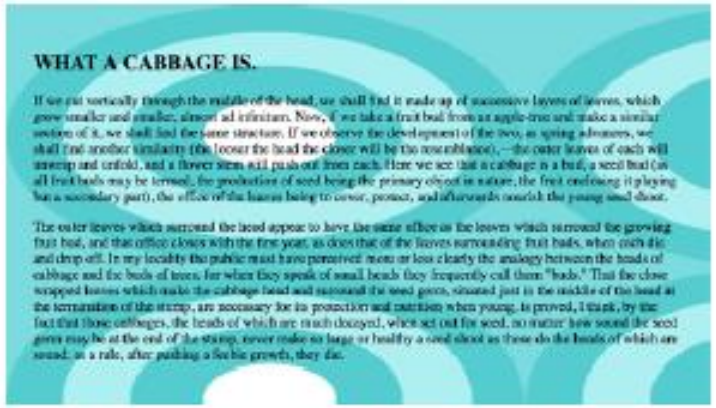
The image is sized proportionally so it fits entirely in the element. There may be room left over for tiling (as shown).

FIGURE 13-28. Examples of the **cover** and **contain** background size keywords.

Background Size

```
header {
    background-image: url(images/target.png);
    background-size: 600px 150px;
}
header {
    background-image: url(images/target.png);
    background-size: 50% 10em;
}
```

target.png
300 × 300 pixels



background-size: 600px 300px;

background-size: 50% 10em;

FIGURE 13-27. Resizing a background image with specific length units and percentages.

Background Size

Multiple Backgrounds

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

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

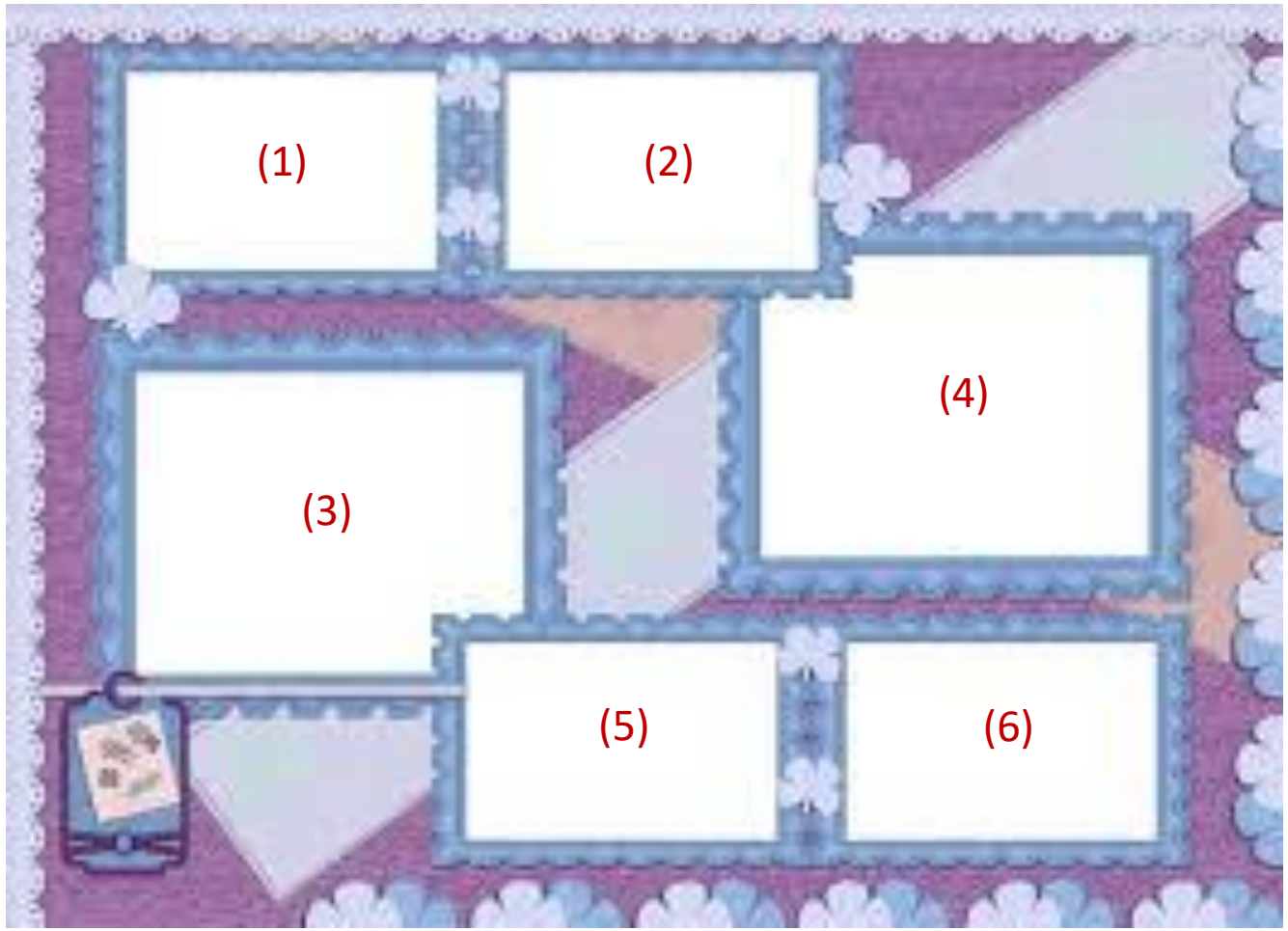
Multiple Backgrounds

FIGURE 13-29. Three separate background images added to the **body** element.

Multiple Backgrounds

Virtual Photo Frame:

```
body {
      background-image: url(frame1), url(frame2),
      url(frame3), url(frame4), url(frame5), url(frame6);
      background-position: 5% 0%, 40% 0%, 5% 50%,
      55% 40%, 40% 75%, 65% 75%
}
```



background style overrides

Only specific style setting can override a previously assigned setting.

Example:

```
h1, h2, h3 { background: red url(dots.gif) repeat-x; }
h3 { background: green; } # This won't override the
                          # first style setting, because
                          # it is not a
                          # specific style setting like
                          # background-color: green;
```

Gradients

SECTION 14

Gradient

- A gradient is a transition from one color to another, sometimes through multiple colors. In the past, the only way to put a gradient on a web page was to create one in an image-editing program and add the resulting image with CSS.
- Now we can specify color gradients by using CSS notation alone, leaving the task of rendering color blends to the browser. Although they are specified with code, gradients are *images*. They just happen to be generated on the fly.
- A gradient image has no intrinsic size or proportions; the size matches the element it gets applied to. Gradients can be applied anywhere an image may be applied: **background-image**, **border-image**, and **list-style-image**. We'll stick with **background-image** examples in this chapter.

Gradient

There are two types of gradients:

- Linear gradients change colors along a line, from one edge of the element to the other.
- Radial gradients start at a point and spread outward in a circular or elliptical shape.

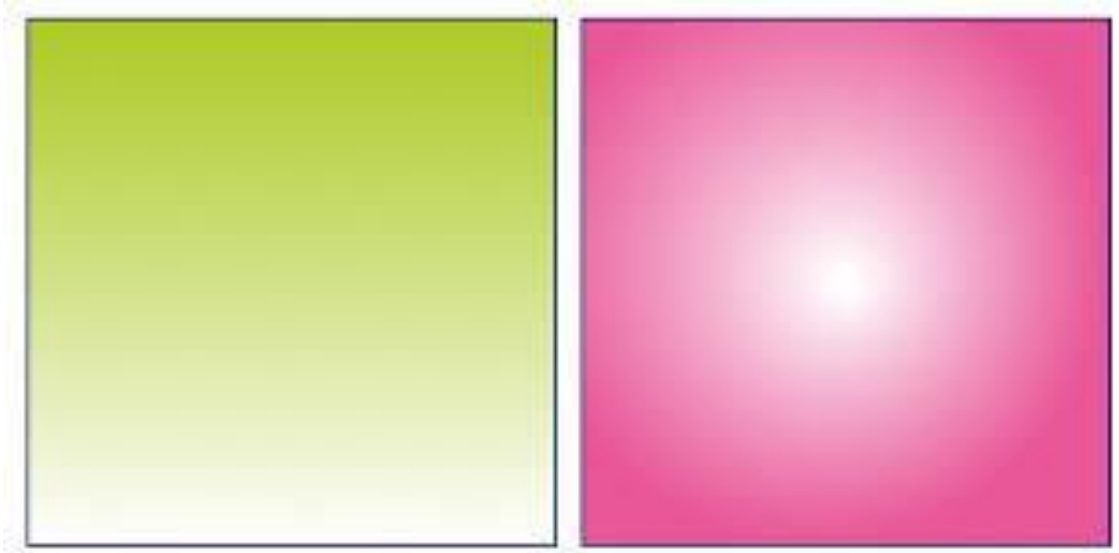
Gradients

Linear Gradients:

```
background: linear-gradient(angle,
            color-stop1, color-stop2);
```

Radial Gradients:

```
background: radial-gradient(
            shape size at position,
            start-color,
            ...,
            last-color);
```

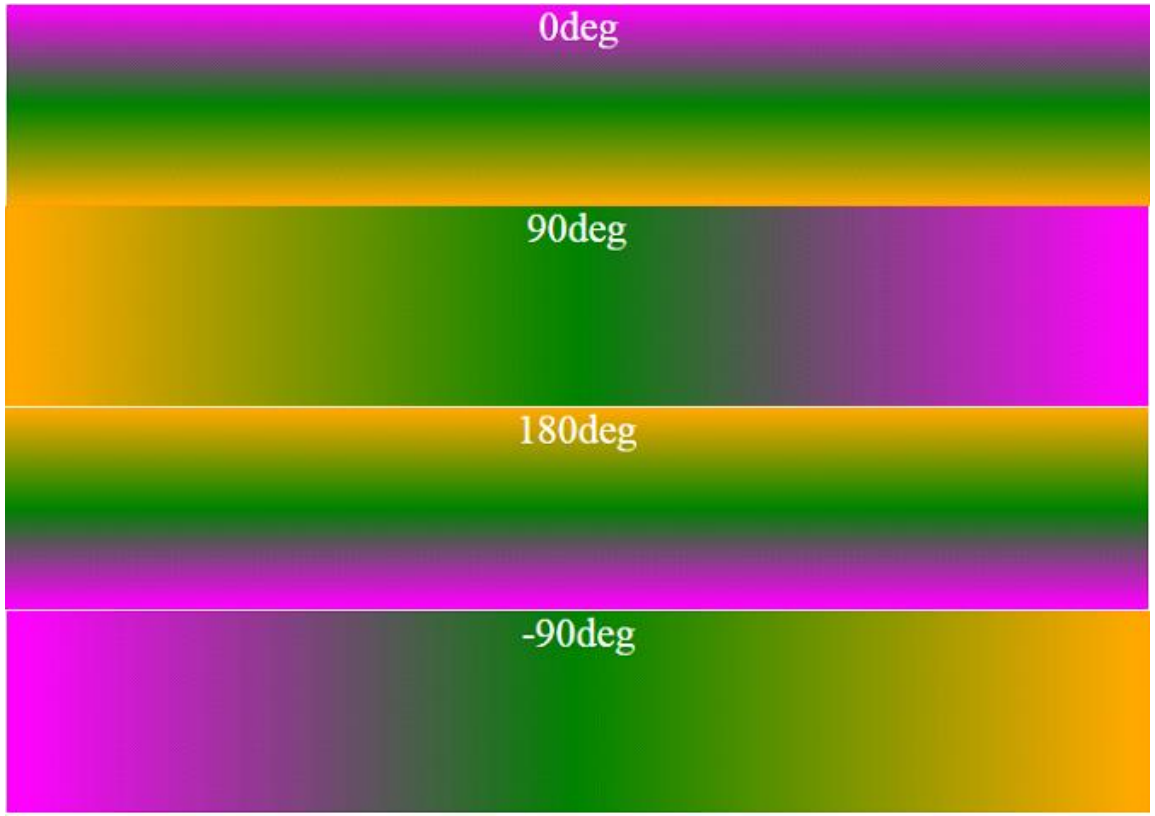


linear-gradient radial-gradient

Linear Gradients

- The **linear-gradient()** notation provides the angle of the gradient line and one or more points along that line where the pure color is positioned (color stops). You can use color names or any of the numerical color values discussed earlier in the chapter, including transparency. The angle of the gradient line is specified in degrees (*n***deg**) or with keywords. With degrees, **0deg** points upward, and positive angles go around clockwise so that **90deg** points to the right. Therefore, if you want to go from aqua on the top edge to green on the bottom edge, set the rotation to **180deg**:

```
background-image: linear-
gradient(180deg, aqua, green);
```

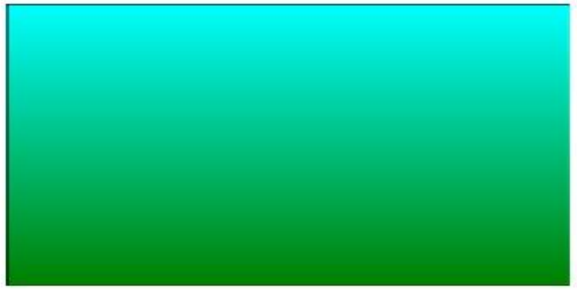


Angle Gradients:

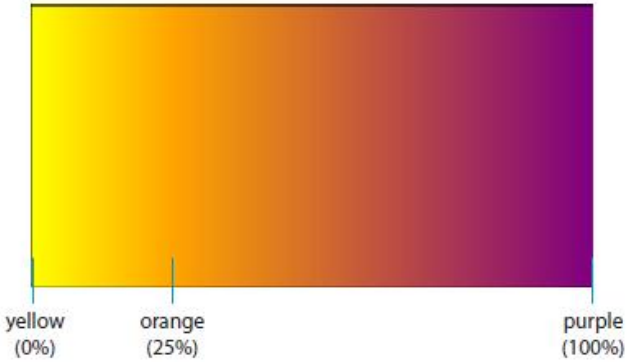
```
#grad_0deg
{
    height: 100px;
    background: linear-gradient(0deg, orange, green, magenta);
}
#grad_90deg
{
    height: 100px;
    background: linear-gradient(90deg, orange, green, magenta);
}
#grad_180deg
{
    height: 100px;
    background: linear-gradient(180deg, orange, green, magenta);
}
```

Linear Gradient

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



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




```
linear-gradient(54deg, red, orange, yellow, green, blue 50%);
```

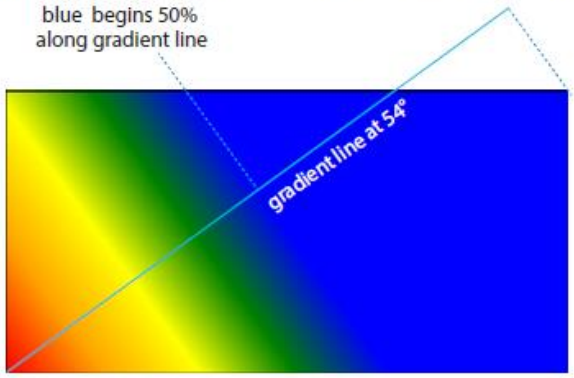


FIGURE 13-31. Examples of linear gradients.

```
#linear_grad
{
    height: 100px;
    background: linear-gradient(yellow, green);
}
```

Linear Gradients

Top to Bottom

```
#linear_grad
{
    height: 100px;
    background: linear-gradient(to right,
    yellow, green);
}
```

Linear Gradient

Left to Right

```
#grad_diagonal
{
    height: 100px;
    background: linear-gradient(to bottom right,
    red ,yellow, green);
}
```

Linear Gradient

Diagonal

```
#grad_transparency
{
    height: 200px;
    background: linear-gradient(to left,
    rgba(100,50,20,0), rgba(100,50,20,1));
}
```

Linear Gradient

Transparency

Radial Gradients

- Radial gradients, like the name says, radiate out from a point in a circle along a gradient ray (like a gradient line, but it always points outward from the center). At minimum, a radial gradient requires two color stops, as shown in this example:

```
background-image: radial-
gradient(yellow, green);
```

- By default, the gradient fills the available background area, and its center is positioned in the center of the element (FIGURE 13-33). The result is an ellipse if the containing element is a rectangle and a circle if the element is square.

Radial Gradients

That looks pretty spiffy already, but you don’t have to settle for the default. The **radial-gradient()** notation allows you to specify the shape, size, and center position of the gradient:

Shape

In most cases, the shape of the radial gradient will result from the shape of the element or an explicit size you apply to it, but you can also specify the shape by using the **circle** or **ellipse** keywords. When you make a gradient a **circle** (without conflicting size specifications), it stays circulareven when it is in a rectangular element (FIGURE 13-34, top).

```
background-image: radial-gradient(circle, yellow, green);
```

Radial Gradients

Size

The size of the radial gradient can be specified in length units or percentages, which apply to the gradient ray, or with keywords. If you supply just one length, it is used for both width and height, resulting in a circle. When you provide two lengths, the first one is the horizontal measurement and the second is vertical (FIGURE 13-34, middle). For ellipses, you can provide percentage values as well, or mix percentages with length values.

background-image: radial-gradient(200px 80px, aqua, green);

Radial Gradients

Size

The size of the radial gradient can be specified in length units or percentages, which apply to the gradient ray, or with keywords. If you supply just one length, it is used for both width and height, resulting in a circle. When you provide two lengths, the first one is the horizontal measurement and the second is vertical (FIGURE 13-34, middle). For ellipses, you can provide percentage values as well, or mix percentages with length values.

background-image: radial-gradient(200px 80px, aqua, green);

There are also four keywords—**closest-side**, **closest-corner**, **farthestside**, and **farthest-corner**—that set the length of the gradient ray relative to points on the containing element.

Radial Gradients

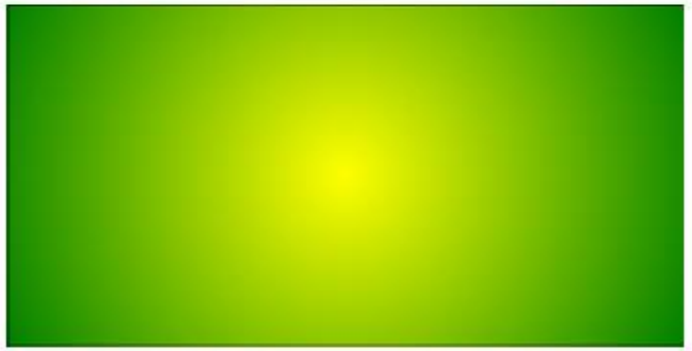
Position

By default, the center of the gradient is positioned at **center center**, but you can change that by using the positioning syntax we covered for the **background-position** property. The syntax is the same, but it should be preceded by the **at** keyword, as in this example (FIGURE 13-34, bottom).
Notice that in this example, I have included an additional color stop of orange at the 50% mark.

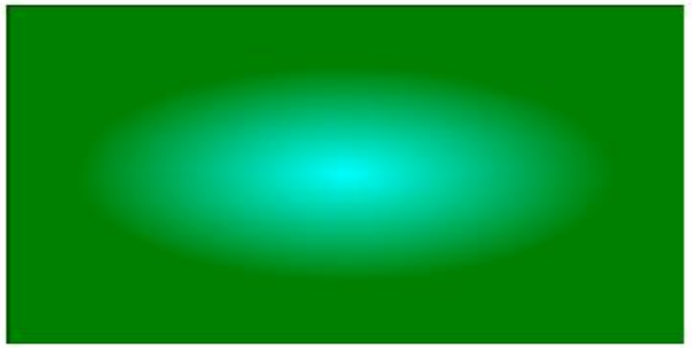
```
background-image: radial-gradient(farthest-side at right bottom,
yellow, orange 50%, purple);
```

Radial Gradients

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



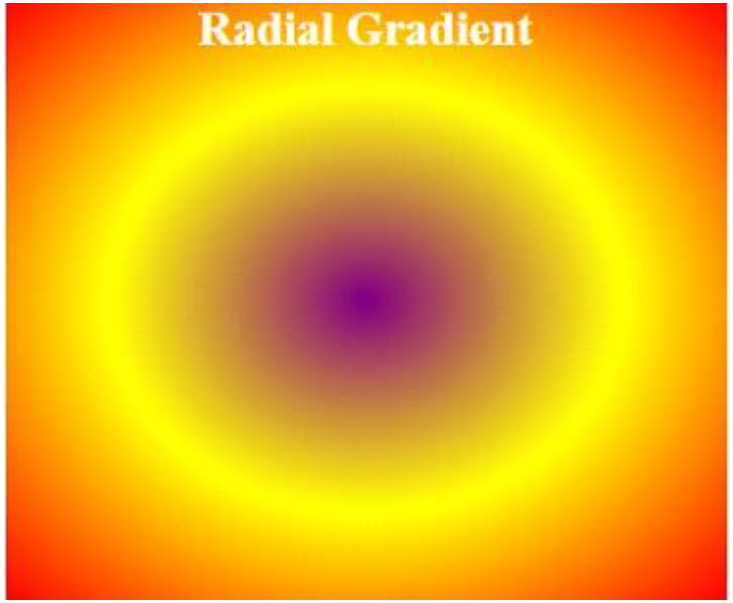
```
radial-gradient(200px 80px, aqua, green);
```



```
radial-gradient(farthest-side at right bottom, yellow, orange 50%, purple);
```

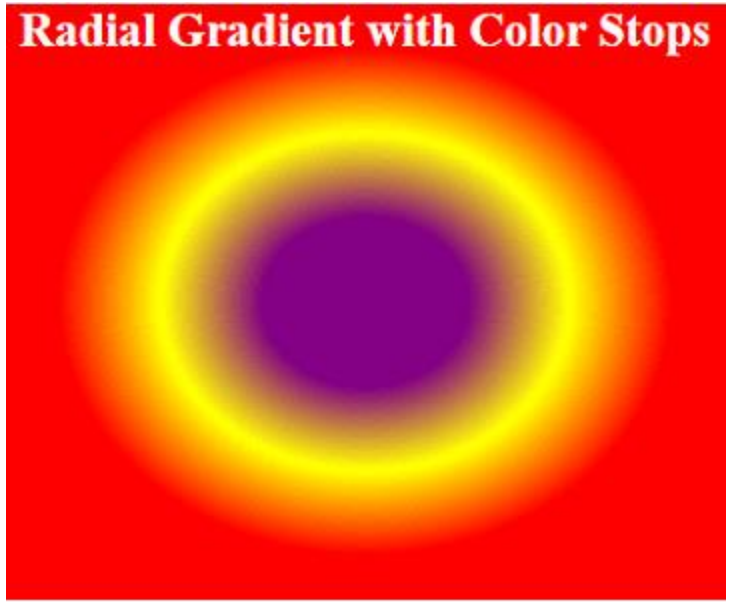
FIGURE 13-34. Examples of sizing and positioning radial gradients.

```
#grad_radial
{
    height: 150px;
    width: 200px;
    background: radial-gradient(purple, yellow,
    red);
}
```



Radial Gradients:

```
#grad_colorstop
{
    height: 250px;
    width: 300px;
    background: radial-gradient(purple 20%,
    yellow 40%, red 60%);
}
```



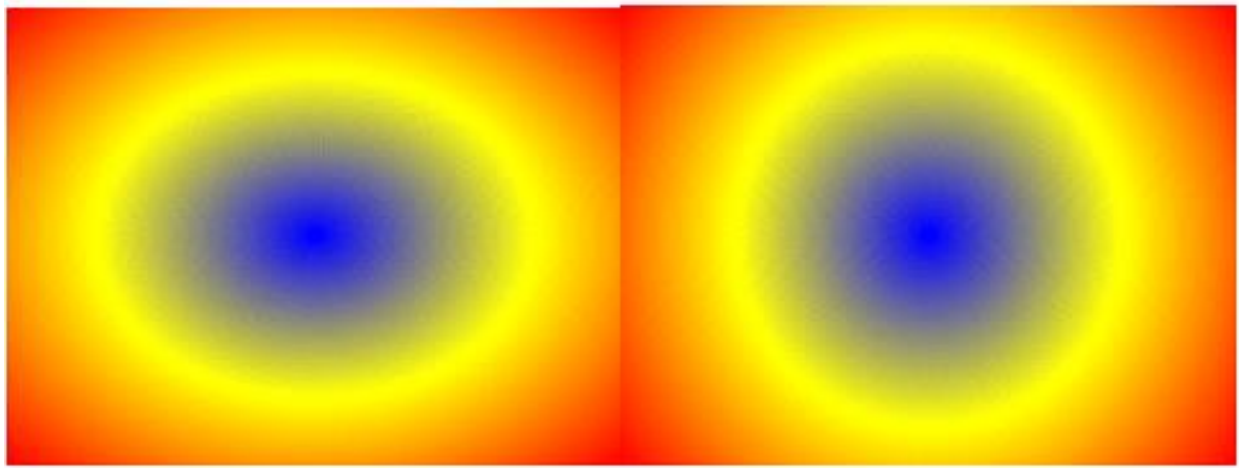
Radial Gradient

Differently Spaced Color Stops

```
#grad_ellips
{
    height: 150px;
    width: 200px;
    background: radial-gradient(blue, yellow, red);
}

#grad_circle
{
    height: 150px;
    width: 200px;
    background: radial-gradient(circle, blue, yellow,
    red);
}
```

Ellipse (this is default): **Circle:**



Set Radial Shapes:

Size Parameter:

The size parameter defines the size of the gradient.
The size parameter includes four values:

- closest-side
- farthest-side
- closest-corner
- farthest-corner

The size parameter defines the size of the gradient. The size parameter includes four values:

- closest-side
- farthest-side
- closest-corner
- farthest-corner

```
#grad1 { height: 150px;
width: 150px;
background: radial-gradient(closest-side at 60% 55%,orange,
yellow,green,yellow, blue); }
#grad2 { height: 150px;
width: 150px;
background: radial-gradient(farthest-side at 60% 55%,orange,
yellow,green,yellow, blue); }
#grad3 { height: 150px;
width: 150px;
background: radial-gradient(closest-corner at 60% 55%,orange,
yellow,green,yellow, blue);}
#grad4 { height: 150px;
width: 150px;
background: radial-gradient(farthest-corner at 60%
55%,orange, yellow,green,yellow, blue);}
```

Repeating Gradients

- If you'd like your gradient pattern to repeat, use the repeating-linear-gradient() or repeating-radial-gradient() notation. The syntax is the same as for single gradients, but adding "repeating-" causes the pattern to repeat the color stops infinitely in both directions. This is commonly used to create interesting striped patterns. In this simple example, a gradient from white to silver (light gray) repeats every 30 pixels because the silver color stop is set to 30px (FIGURE 13-35, top):

```
background: repeating-linear-gradient(to bottom, white, silver 30px);
```

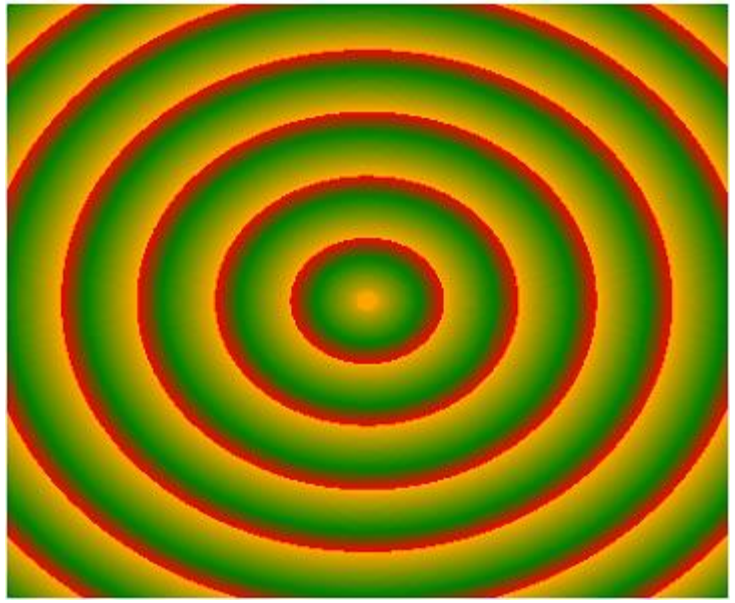
Repeating Gradients

```
repeating-linear-gradient(to bottom, white, silver 30px);
```

```
repeating-linear-gradient(45deg, orange, orange 12px, white 12px, white 24px);
```

FIGURE 13-35. Repeating gradient pattern.

```
#grad1
{
    height: 250px;
    width: 300px;
    background: repeating-radial-
    gradient(orange, green 10%, red 15%);
}
```



Repeating a radial-gradient:

Conclusion for Gradients:

- This part is aimed towards adding background gradients to any HTML element. Just follow the example and demo to add background image to any HTML element.
- This tutorial shows how the background gradient works on different HTML elements. For using background gradients we can create an attractive background by using different colors.

Vendor-dependent Issues

SECTION 15

Vendor prefixes

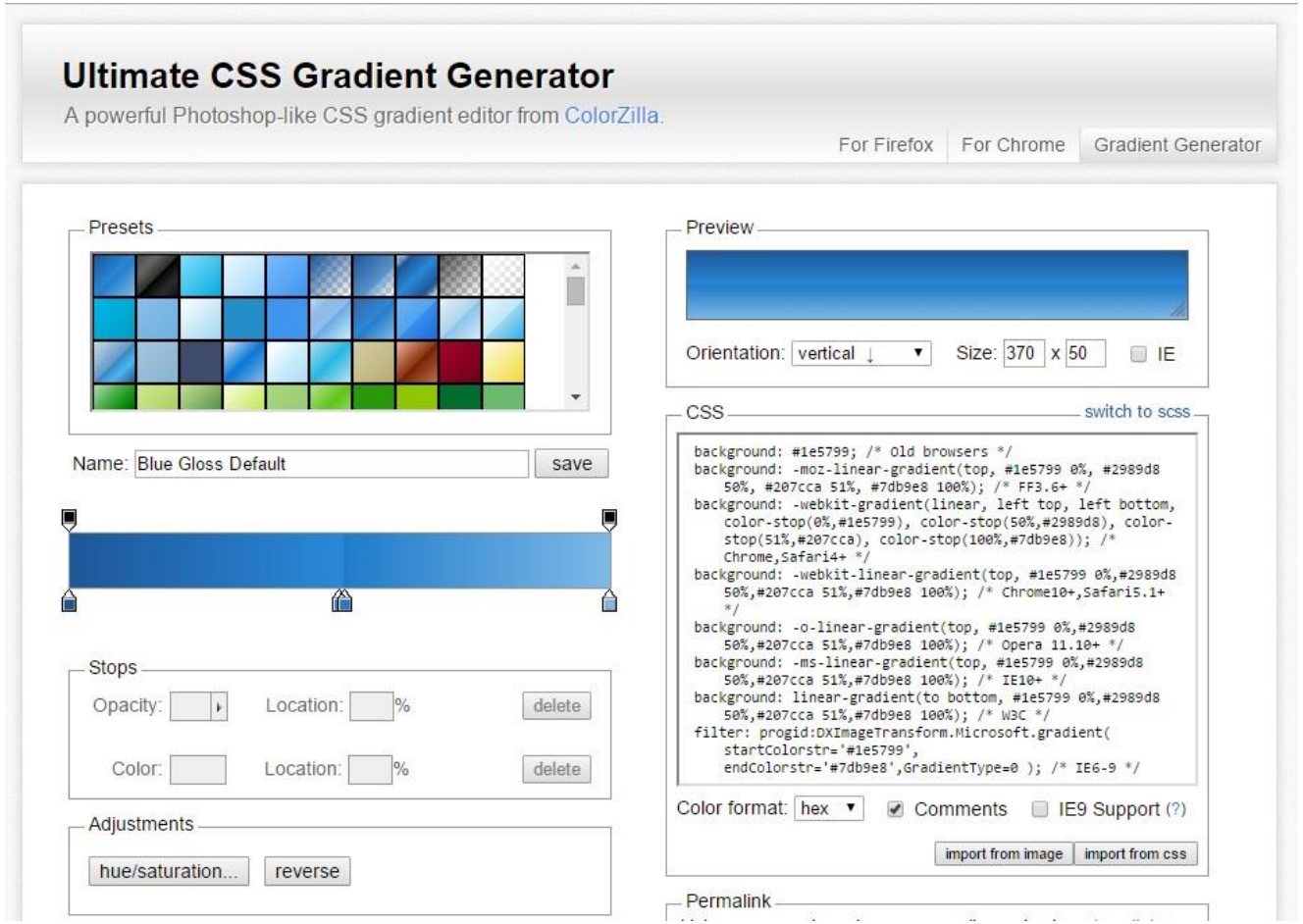
Prefix	Organization	Most Popular Browsers
-ms-	Microsoft	Internet Explorer
-moz-	Mozilla Foundation	Firefox, Camino, SeaMonkey
-o-	Opera Software	Opera, Opera Mini, Opera Mobile
-webkit-	Originally Apple; now open source	Safari, Chrome, Android, BlackBerry, WebOS, Many Others
-khtml-	Konqueror	Konqueror

What vendor prefixes are for?

Vendor prefixes are for designers and developers to add newer CSS3 features, to create more function or features for users.

For example,

```
background: -webkit-gradient(linear, left top, left bottom, color-stop(#ffff00), color-stop(#00ff00));
```



Free Gradient Designer
www.colorzilla.com/gradient-editor