WEB SYSTEM AND TECHNOLOGIES

Chapter 3
CSS
Cascading Style Sheet

Content

1. What is CSS?
2. Styling with Cascading Stylesheets (CSS)
3. Selectors and style definitions
4. Linking HTML and CSS
5. Fonts, Backgrounds, Borders
6. The Box Model
7. Alignment, z-Index, Margin, Padding
8. Positioning and Floating Elements
9. Visibility, Display, Overflow

Before CSS

```
<body bgcolor="orange">
    <h1 font-size="34px">This is a heading.</h1>
    <div bgcolor="red">
      <p>
          <i><b>This is a paragraph.</b></i>
      </p>
     </div>
</body>
```

- Before had CSS all style information was encoded directly into HTML documents.

After CSS

example.html

```
<head>
    <link rel="stylesheet" type="text/css"
href="style.css">
</head>
<body>
    <h1>This is a heading.</h1>
    <div>
      <p>This is a paragraph</p>
    </div>
</body>
```

style.css

```
body { background-color: orange; color: blue;}
h1 { font-size: 34px; }
div { background-color: red; }
p { font-weight: bold; font-style: italic;}
```

What is CSS? A new Philosophy

Content
(HTML document)

Presentation
(CSS Document)

Separate content from presentation!

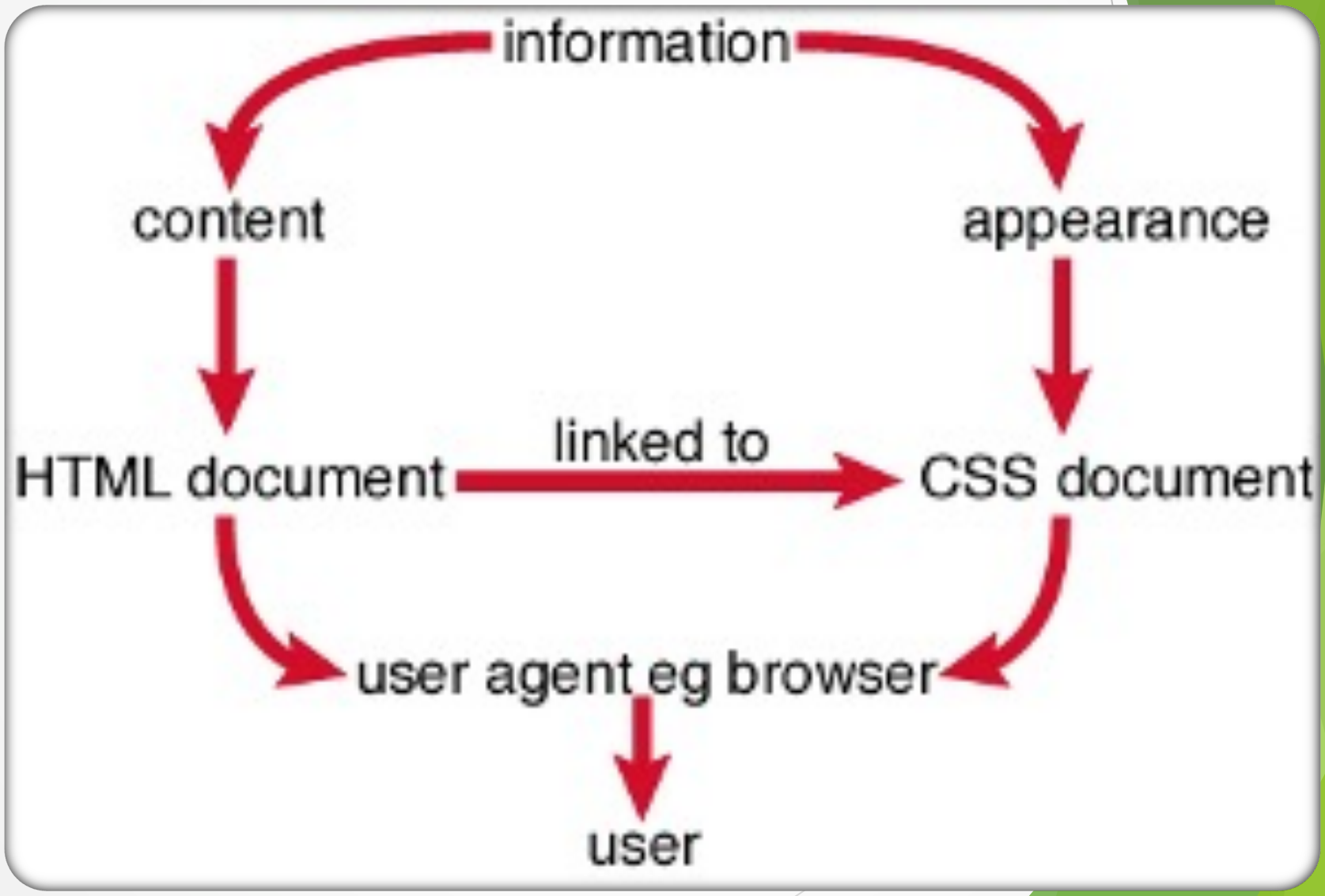
Title

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Suspendisse at pede ut purus malesuada dictum. Donec vitae neque non magna aliquam dictum.

- Vestibulum et odio et ipsum
- Accumsan accumsan. Morbi at
- Arcu vel elit ultricies porta. Proin

tortor purus, luctus non, aliquam nec, interdum vel, mi. Sed nec quam nec odio lacinia molestie. Praesent augue tortor, convallis eget, euismod nonummy, lacinia ut, risus.

How do Style Sheets work?



CSS Introduction

- What is CSS?
  - Is a language used to specify formatting of an HTML document
  - Have a large set of style attributes that can be used to style HTML elements
- Why CSS?
  - Because it helps to separate content of an HTML document from its presentation

Advantages of CSS

- CSS saves time:
  - Can write CSS once and the reuse same sheet in multiple HTML pages.
- Pages load faster:
  - If you are using CSS, you do not need to write HTML tag attributes every time, less code means faster download times.
- Easy maintenance:
  - To make a global change, simply change the style and all elements in all the web pages will be updated automatically.

Advantages of CSS (cont.)

- Superior style to HTML:
  - CSS has a much wider array of attributes than HTML
- Multiple device compatibility:
  - Style sheets allow content to be optimized for more than one type of device. By using the same HTML document, different versions of a website can be presented for handheld devices such as PDAs and cell hones or for printing.

CSS History

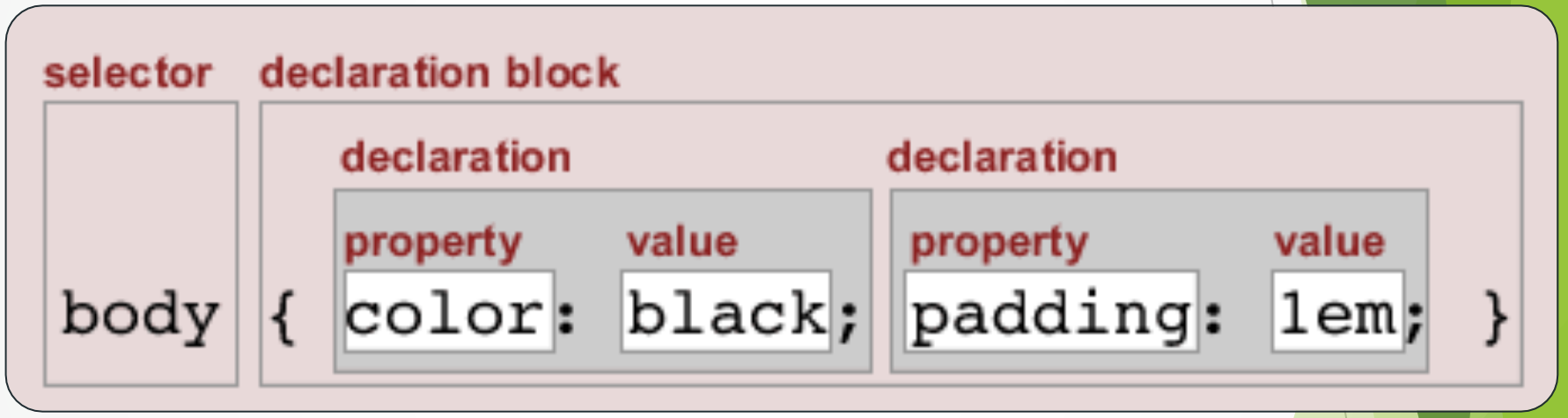
- Cascading Style Sheets, level 1 (CSS1) was came out of W3C as a recommendation in December 1996
- CSS2 was became as W3C recommendation in May 1998 and builds on CSS1
- CSS3 was became a W3C recommendation in June 1999 and builds on older versions CSS

Why “Cascading”?

- Some CSS styles are inherited and some not
  - Text-related and list-related properties are inherited - `color, font-size, font-family, line-height, text-align, list-style,…`
  - Box-related and positioning styles are not inherited - `width, height, border, margin, padding, position, float,…`
  - `<a>` elements do not inherit color and text-decoration

CSS Syntax

- Stylesheets consist of rules, selectors, declarations, properties and values



- Selectors are separated by commas
- Declarations are separated by semicolons
- Properties and values are separated by colons

```
h1,h2,h3 { color: green; font-weight: bold; }
```

Linking HTML and CSS

HTML (content) and CSS (presentation) can be linked in three ways:

- **Inline**: the CSS rules in the style attribute
  - No selectors are needed
- **Internal**: in the `<head>` in a `<style>` tag [.html]
- **External**: CSS rules in separate file (***BEST***)
  - Usually a file with *.css* extension
  - Linked via `<link rel="stylesheet" type="text/css" href="…">` tag or `@import` directive in embedded CSS block

Inline style

- An inline style may be used to apply a unique style for a single element.
- The style attribute can contain any CSS property.

```
...
<body>

<h1 style="color:blue;text-
align:center;">This is a heading</h1>
<p style="color:red;">This is a
paragraph.</p>

</body>
...
```

Internal style

- Internal style in the HTML in the <style> tag:

```
<style type="text/css">
</style>
```

  - The <style> tag is placed in the <head> section of the document
  - Type attribute specifies the MIME type
    - MIME describes the format of the content
    - Other MIME types include text/html, image/gif, text/javascript …
- Used for document-specific styles

External style

- External linking
  - Separate pages can all use a shared style sheet
  - Only modify a single file to change the styles across your entire Web site link tag (with a rel attribute)
  - Specifies a relationship between current document and another document link elements should be in the <head>

```
<link rel="stylesheet" type="text/css"
href="styles.css">
```

External style (cont.)

@import

- The @import rule allows you to import a style sheet into another style sheet.
- The @import rule must be at the top of the document (but after any @charset declaration).
- The @import rule also supports media queries, so you can allow the import to be media-dependent.

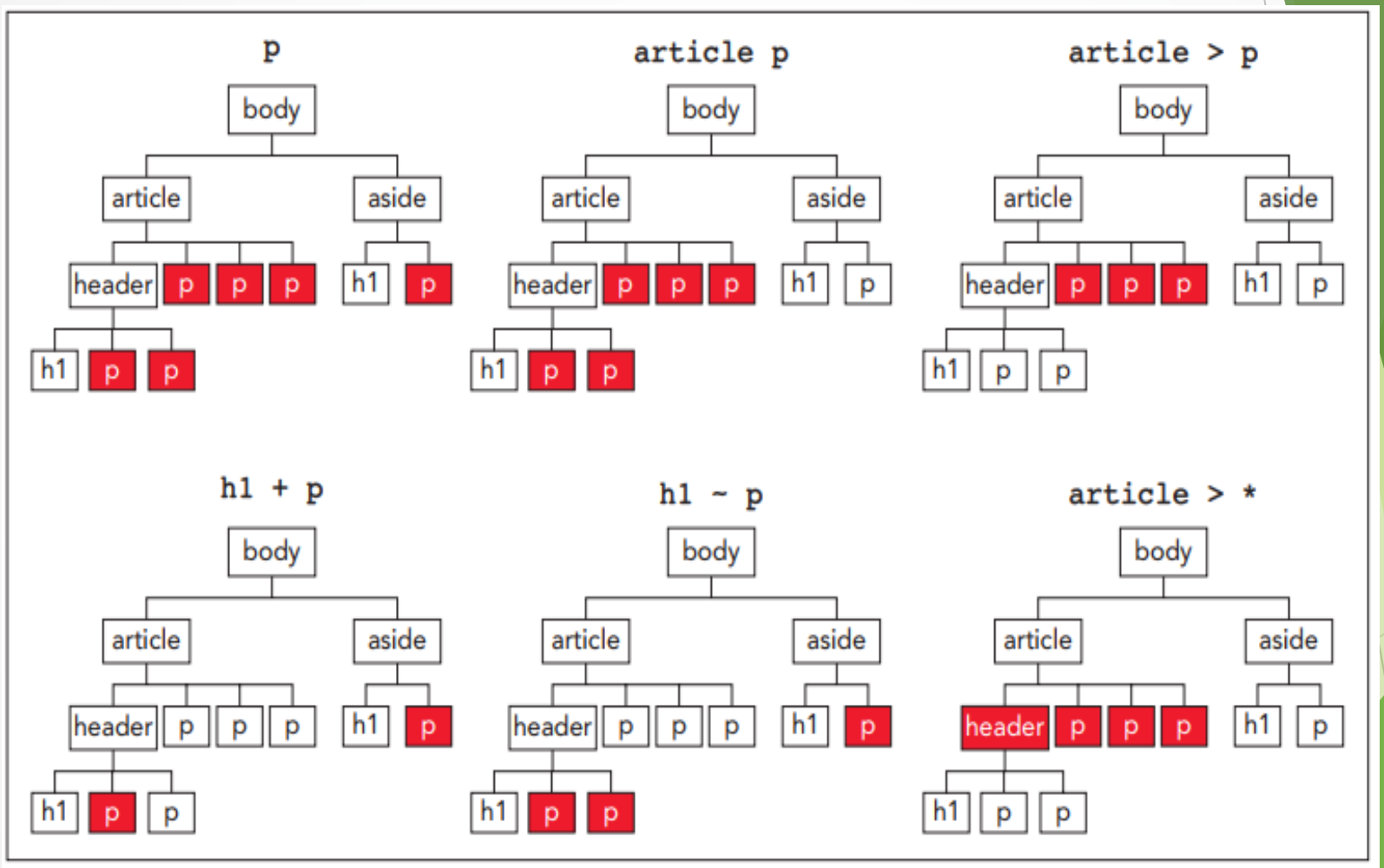
```
<style type="text/css">
  @import url("styles.css");
  @import "printstyle.css" print;
</style>
```

External style: Example

styles.css

```
/* CSS Document */

a        { text-decoration: none }

a:hover { text-decoration: underline;
          color: red;
          background-color: #CCFFCC }

li em    { color: red;
          font-weight: bold }

ul       { margin-left: 2cm }

ul li    { text-decoration: underline;
          margin-left: .5cm }
```

Relationship Selectors



Selector

- Selectors determine which element the rule applies to:
  - All elements of specific type (tag)
  - Those that match a specific attribute (id, class)
  - Elements may be matched depending on how they are nested in the document tree (HTML)
- Examples:

```
.header a { color: green }

#menu>li { padding-top: 8px }
```

Type of Selectors

- The Element selector: used to select an HTML element by its tag name

```
h1 { color:green; }
```

- The ID selector: used to select an HTML element by its id

```
#element_id { color:green; }
```

- The Class Selector
  - Define generic styles that can be applied to different HTML elements

```
.myClass { color:green; }
```

  - To select all p elements found inside a given class

```
p.myClass { color:red; }
```

Type of Selectors (cont.)

- The Universal Selectors: matches the name of any element type

```
* { color:green; }
```

- The Attributes selectors: to select HTML elements with particular attributes

```
input[type] { color:green; }
input[type="text"] { color:green; }
```

- Grouping Selectors: to apply style to many selectors, just separate the selectors with a comma

```
h1,h2,h3,#para,.myClass { color:green; }
```

Selector (cont.)

- Descendant Selector: match relative to element placement:

```
p a {text-decoration: underline;}
```

- This will match all <a> tags that are inside of <p>
- General Sibling Selector: select all elements that are siblings of a specified element

```
p ~ a {text-decoration: underline;}
```

- Select all <a> elements that are siblings of <p> elements
- Adjacent Sibling Selector: + selector – used to match “next sibling”:

```
p + a {text-decoration: underline;}
```

- This will match all siblings with <a> tag that appear immediately after <p> tag

Selector (cont.)

- Child Selector: > selector – matches direct child nodes:

```
p > .error {font-size: 8px}
```

  - This will match all elements with class error, direct children of <p> tag
- [ ] – matches tag attributes by regular expression:

```
img[alt~=logo] {border: none}
```

  - This will match all <img> tags with alt attribute containing the word logo
- .class1.class2 (no space) - matches elements with both (all) classes applied at the same time

Example

```
<!DOCTYPE html>
<html>
<head>
  <title>Selector</title>
  <style type="text/css">
      div > span { color:red; } /* Child Selector */
      div span { color:blue; } /* Descendant Selector */
      div > p { color:red; } /* Child Selector */
      h1 + h2 { color:yellow; } /* Adjacent Sibling Selector */
      h1 ~ h2 { color:red; } /* Sibling Selector */
      h1 + a { color:yellow; } /* Adjacent Sibling Selector */
  </style>
</head>
<body>
  <div>
    <h1>Heading One</h1>
    <h2>Heading Two</h2>
    <p>Paragraph One</p>
    <p>Paragraph <span>Two</span></p>
    <a href="#">Link One</a>
    <a href="#">Link Two</a>
  </div>
</body>
</html>
```

Pseudo Selector

- Pseudo-class is used to define a special state of an element
- For example, it can be used to:
  - Style an element when a user mouses over it
  - Style visited and unvisited link differently
- Some CSS Pseudo Classes:

Pseudo Selector (cont.)

Selector	Example	Description
:first-child	p:first-child	Selects every <p> elements that is the first child of its parent
:last-child	p:last-child	Selects every <p> elements that is the last child of its parent
:not(selector)	:not(p)	Selects every element that is not a <p> element
:nth-child(n)	p:nth-child(2)	Selects every <p> element that is the second child of its parent
:empty	p:empty	Selects every <p>element that has no children
:optional	input:optional	Selects *<input>* elements with no "required" attribute
:hover	div:hover	div:hover { background-color: blue; }

Pseudo Selector (cont.)

- A simple syntax of pseudo-classes is as follows:

```
selector:pseudo-class { property:value; }
```

- CSS classes can also be used with pseudo-classes:

```
selector.class:pseudo-class { property:value; }
```

```
a:hover { color: green; }
p:first-line { text-transform: uppercase; }
p.title:before { content: "»"; }
p.title:after { content: "«"; }
```

Standard Pseudo-classes

- :active
- :any
- :checked
- :default
- :dir()
- :disabled
- :empty
- :enabled
- :first-child
- :first-of-type
- :fullscreen
- :focus
- :hover
- :indeterminate
- :in-range
- :invalid
- :lang()
- :last-child
- :last-of-type
- :left
- :link
- :not()
- :nth-child()
- :nth-last-child()
- :nth-last-of-type()
- :nth-of-type()
- :only-child
- :only-of-type
- :read-only
- :read-write
- :right
- :scope
- :target
- :valid
- :visited

Pseudo Selector (cont.)

```
<!DOCTYPE html>
<html>
<head>
  <title>Pseudo Selector</title>
  <style type="text/css">
      p {
          display: none;
          background-color: yellow;
          padding: 20px;
      }
      div:hover p {
          display: block;
      }
  </style>
</head>
<body>
  <div>Hover over me to show the p element
    <p>Tada! Here I am!</p>
</div>
</body>
</html>
```

Pseudo Elements

- Pseudo-elements is used to style specified parts of an element.
- Example, it can be used to:
  - Style the first letter, or line of an element
  - Insert content before or after the content of an element

```
selector::pseudo-element { property:value; }
```

Pseudo Elements (cont.)

- Some CSS Pseudo Elements (The double colon replaced the single colon for pseudo-elements in CSS3 to distinguish pseudo-element)

Selector	Example	Description
::after	p::after	Insert content after every <p> element
::before	p::before	Insert content before every <p> element
::first-letter	p::first-letter	Selects the first letter of every <p> element
::first-line	p::first-line	Selects the first line of every <p> element
::selection	p::selection	Selects the portion of an element that is slected by a user

Pseudo Elements (cont.)

```
<!DOCTYPE html>
<html>
<head>
  <title>Pseudo Selector</title>
  <style type="text/css">
      p::first-line { text-decoration: underline; }
      p.noline::first-line { text-decoration: none; }
      p::first-letter { font-size: 5em; }
      p::before { content: url(/images/bullet.gif); }
      p::after { content: “Thank you”; color: red; font-size: 34px }
  </style>
</head>
<body>
  <div>
    <p>Line 1: First paragraph in div<br>
    Line 2: First paragraph in div</p>
    <p class=“noline”>Second paragraph indic.</p>
  </div>
</body>
</html>
```

Values in the CSS Rules

- Colors are set in RGB format (decimal or hex):
  - Example: `#a0a6aa = rgb(160, 166, 170)`
  - Predefined color aliases exist: black, blue, etc.
- Numeric values are specified in:
  - Pixels, ems: `12px , 1.4em`
  - Points, inches, centimeters, millimeters: `10pt , 1in, 1cm, 1mm`
  - Percentages: `50%`
  - Zero can be used with no unit: `border: 0;`

Default Browser Styles

- Browsers have default CSS styles
  - Used when there is no CSS information or any other style information in the document
- Caution: default styles differ in browsers
  - Ex: margins, paddings and font sizes differ most often and usually developers reset them

```
* {
      margin: 0 auto;
      padding: 0;
      box-sizing: border-box;
   }
```

CSS Cascade (Precedence)

- There are browsers, user and author stylesheets with "normal" and "important" declarations
  - Browser styles (least priority)
  - Normal user styles
  - Normal author styles (external, in head, inline)
  - Important author styles
  - Important user styles (max priority)

```
a { color: red !important ; }
```

CSS Comments

- Comments are ignored by browsers
- *A CSS comment start with /* and ends with */*

  Comments can also span multiple lines

```
p {
      color: red;
      /* This is a single-line comment */
      text-align: center;
   }
```

CSS Color

- Colors in HTML can be specified by the following methods:
  - A color name – like: green
  - A HEX value in a format #RRGGBB – like: #ff0000
    - Hexadecimal values between 00 and FF
  - An RBG value – like: rbg(255,255,255)

```
p { color: green; }
h1 { color: rgb(0,255,0); }
div { color: #008000 }
```

CSS – Measurement Units

Unit	Description	Example
Pt	Defines a measurement in points. A point is defined as 1/72nd of an inch.	body {font-size: 18pt;}
Px	Defines a measurement in screen pixels.	img {width: 25px;}
%	Defines a measurement as a percentage relative to another value, typically an enclosing element.	body{width: 800px; } div{width: 20%;}
Cm	Defines a measurement in centimeters.	div {width: 2cm;}
In	Defines a measurement in inches.	p {word-spacing: .15in;}
Mm	Defines a measurement in millimeters.	p {word-spacing: 15mm;}
Em	Relative to the font-size of the element (2em means 2 times the size of the current font)	p {font-size: 2em;}

CSS Font

- Following font properties of an element
  - font-family: is used to specify font face
  - font-style: is used to make a font italic or oblique
  - font-variant: is used to create a small-caps effect
  - font-weight can be normal, bold, bolder, lighter or a number in range [100 … 900]
  - font-size – size of font: `xx-small (9px), x-small (10px), small (13px), **medium (16px)**, large (18px), x-large (24px), xx-large (32px), xxx-large (48px).`
  - font property: as shorthand

CSS Font (cont.)

- font-family property: there are two types of font family names:
  - Generic family: a group of font families with a similar look (like Serif or Monospace)
  - Font family: a specific font family (like Times New Roman or Arial)

Generic Family	Font Family
Serif	Times New Roman Georgia
Sans-serif	Arial Verdana
Monospace	Courier New Lucida Console

CSS Font (cont.)

Font Properties	Example	Possible Values
font-family	font-family: "Times New Roman", Times, serif;	
font-style	font-style:italic;	*normal* *italic* *oblique*
font-variant	font-variant:small-caps;	*small-caps* *normal*
font-weight	font-weight:bold;	*normal, bold, bolder, lighter, 100, 200, 300, 400, 500, 600, 700, 800, 900*
font-size	font-size:20px;	*xx-small, x-small, small, medium, large, x-large, xx-large, smaller, larger, size in pixels or in %*

CSS3 Web Font @font-face rule

- Web fonts:
  - Allow Web designers to use fonts that are not installed on the user's computer
  - When you found/bought the font you wish to use, just include the font file on your web server and it will be automatically downloaded to the user when needed.
- In CSS3 @font-face rule, you must first define a name for font and then point to the font file.

```
@font-face {
  font-family: myFirstFont;
  src: url(sansation_light.woff);
}
div { font-family: myFirstFont; }
```

CSS Text

- color – specifies: the color of the text
- direction: set the text direction
- letter-spacing: add or subtract space between the letters
- word-spacing: add or subtract space between the words
- text-indent: set indent the text
- text-align: property set the horizontal alignment of a text {left, right, center, or justified}
- vertical-align: {top, bottom, middle}
- line-height: {size px %}
- text-decoration: underline, overline, strikethrought text
- text-shadow: set the text shadow around a text
- text-transform: property controls the capitalization of text.

CSS Text (cont.)

Text Properties	Example	Possible Values
`color`	`color:red;`	
`text-align`	`text-align:right;`	*left, right, center, justify*
`text-decoration`	`text-decoration:underline;`	*none, underline, overline, line-through, blink*
`text-transform`	`text-transform:capitalize;`	*none, capitalize, uppercase, lowercase.*
`text-indent`	`text-indent:1cm;`	*% or a number specifying indent space*
`word-spacing`	`word-spacing:5px;`	*normal or a number specifying space*
`letter-spacing`	`letter-spacing:5px;`	*normal or a number specifying space*
`direction`	`direction:rtl;`	*ltr or rtl*
`text-shadow`	`text-shadow:4px 4px 8px blue;`	

CSS Text Shadow

```
h1 {
   text-shadow: 2px 2px #FF0000;
}
```


```
h1 {
   text-shadow: 0px 2px 2px #ffffff,
                0px -2px 2px #FF0000
}
```

CSS Text Transform

- Move the word “DUKES” over to the right when the user hovers over it.

```
#divExample h1:hover span {
  color: #484848
  -webkit-transform: translate(40px); /*Webkit Chrome &
Safari*/
  -moz-transform: translate(40px);/*Mozilla*/
  -ms-transform: translate(40px); /*IE*/
  -o-transform: translate(40px);/*Opera*/
  transform: translate(40px); /*CSS3*/
}
```

CSS Links

- Links can be styled with any CSS property: color, font-family, background, …

```
a { color: #000; text-decoration: none;}
```

- Links can be styled differently depending on what state they're in:
  - a:link – a normal, unvisited link
  - a:visited – a link the user visited
  - a:hover – when the user mouses over it
  - a:active – the moment it is clicked

```
a:link {color: #000 }
a:visited {color:
#006600}
a:hover {color: #ffcc00}
a:acive {color: #ff00cc}
```

- When setting the style for several link states, there're some order rules:
  - a:hover MUST come after a:link and a:visited
  - a:active MUST come after a:hover

CSS List

Image Properties	Example	Possible Values
list-style-type	list-style-type:decimal; list-style-type:square;	*None, disc, circle, square decimal, lower-alpha, upper-alpha, lower-roman, upper-roman*
list-style-image	list-style-image:url(/images/bullet.gif)	
list-style-position	list-style-position : outside;	*inside, outside*
marker-offset	marker-offset:2em;	
Shorthand property	list-style: square inside url("bullet.jpg");	

CSS Background

- background-image:url('image file')
  - URL of image to be used as background, e.g.:

```
background-image:url("back.gif");
```

- background-color: {color value}
  - Using color and image and the same time
- background-repeat: {repeat-x, repeat-y, repeat, no-repeat}
- background-attachment: {fixed / scroll}
- background-size: {size of image to set up back ground - %, px}

CSS Background (cont.)

- background-position: {top, center, bottom, left, center, right, %, px}

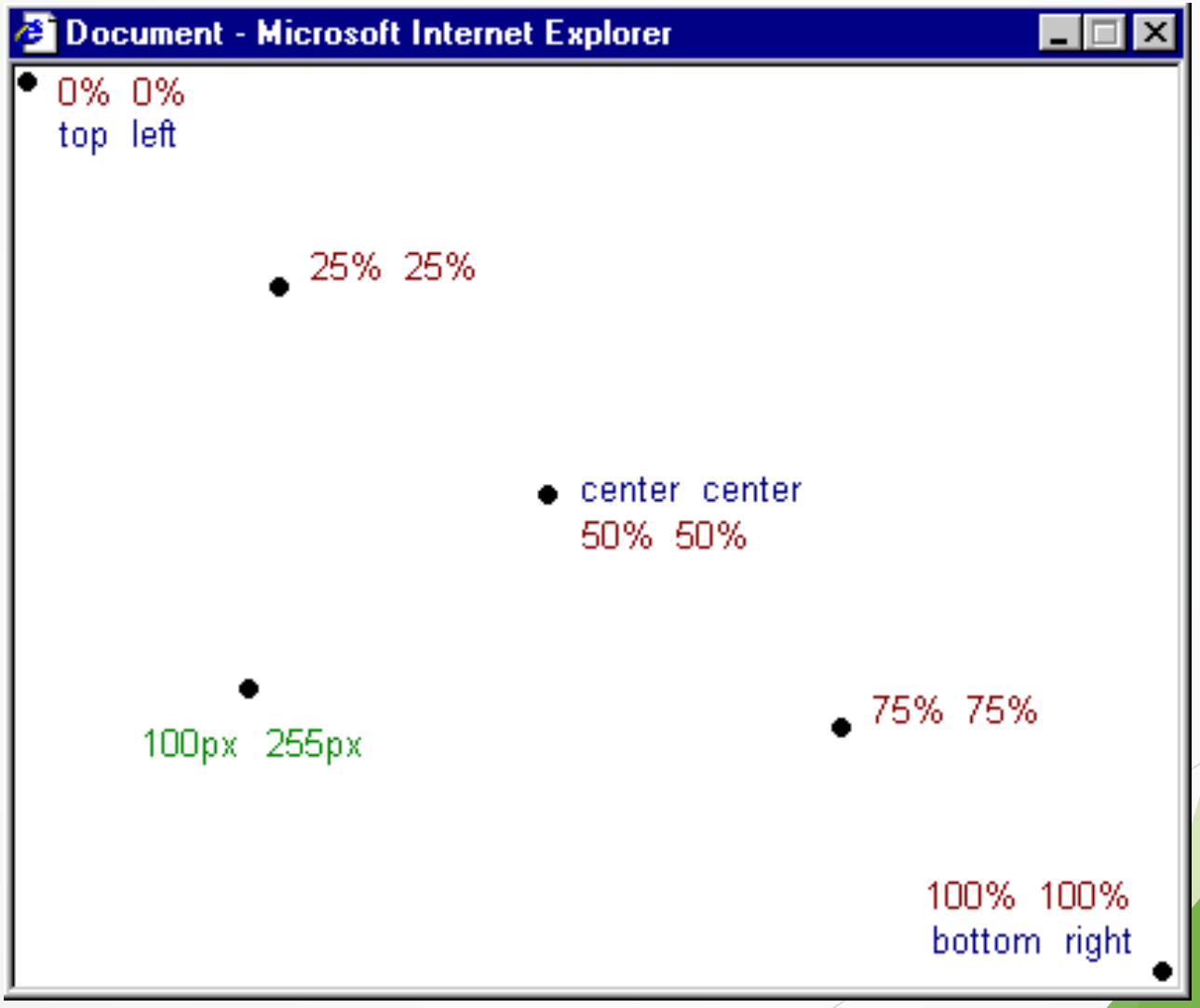
specifies vertical and horizontal position of the background image

Examples:

```
background-position: top left;
```

```
background-position: -5px 50%;
```

CSS Background (cont.)



CSS Background Shorthand

- background: shorthand rule for setting background properties at the same time:

```
background: #FFF0C0 url("back.gif") no-repeat fixed top;
```

is equal to writing:

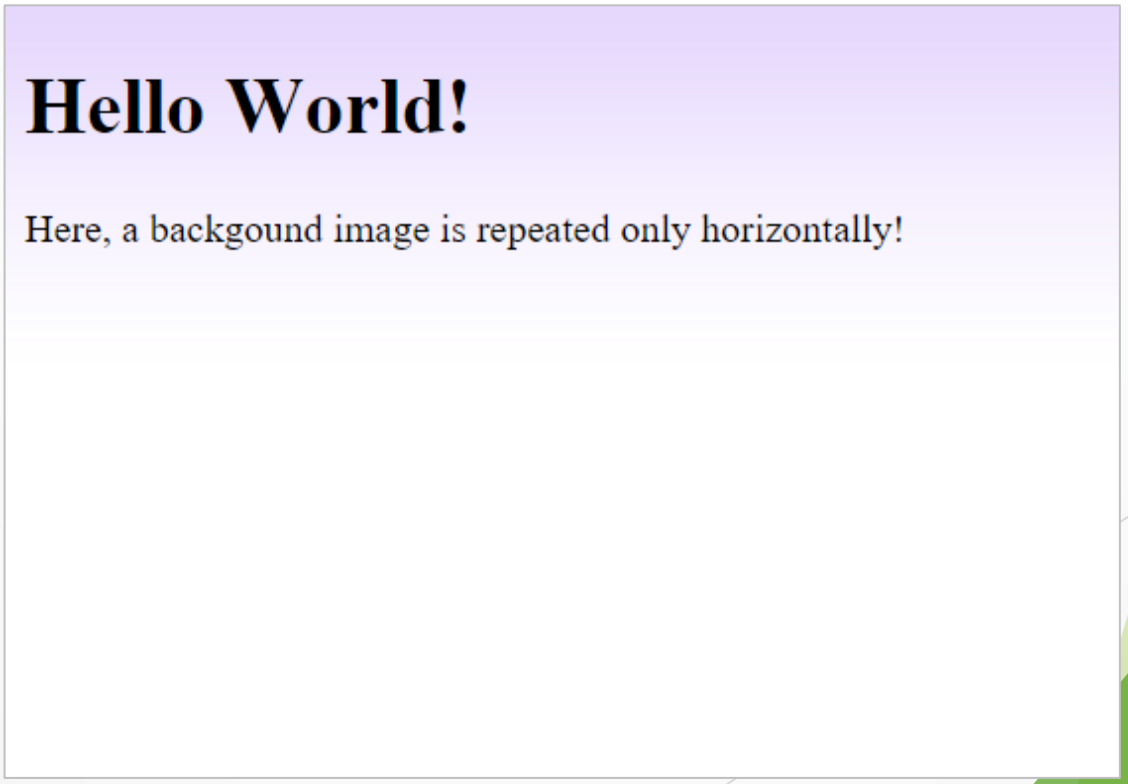
```
background-color: #FFF0C0;
background-image: url("back.gif");
background-repeat: no-repeat;
background-attachment: fixed;
background-position: top;
```

CSS Background Example

```
body {
    background-image: url("gradient_bg.png");
}
```

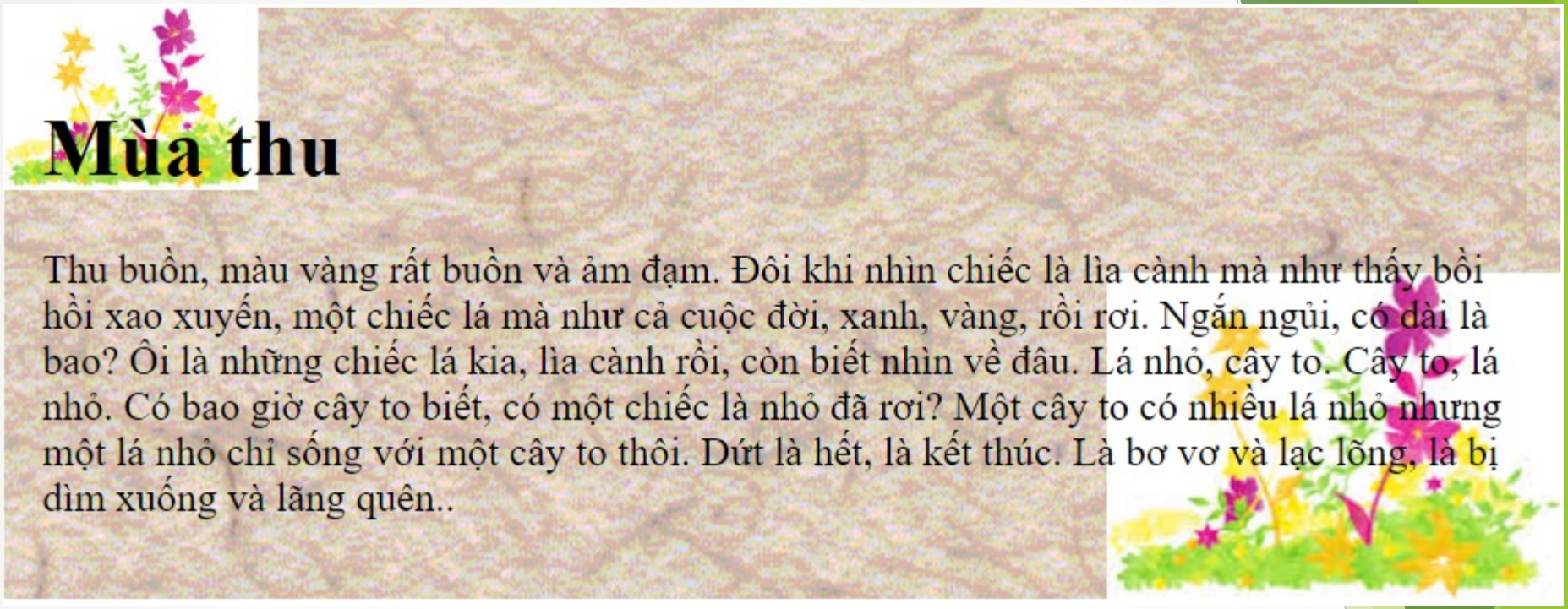
CSS Background Example (cont.)

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



Background-image or <img>?

- Background images allow you to save many image tags from the HTML
  - Leads to less code
  - More content-oriented approach
- All images that are not part of the page content (and are used only for "beautification") should be moved to the CSS




```
#example1 {
    background: url(img_flwr.gif) left top no-repeat,
url(img_flwr.gif) right bottom no-repeat, url(paper.gif) left top
repeat;
    background-size: 50px, 130px, auto;
}
```

CSS Menu Example

- Home
- News
- Contact
- About


```
<style>
  ul {padding-left: 0;}
  ul li {
    list-style-type: none;
    width: 150px;
    border: 1px solid #fff;
    display: block;
    background-color: #000;
    text-align: center;
    color: #fff;
    padding: 20px;
  }
</style>
```

CSS Border

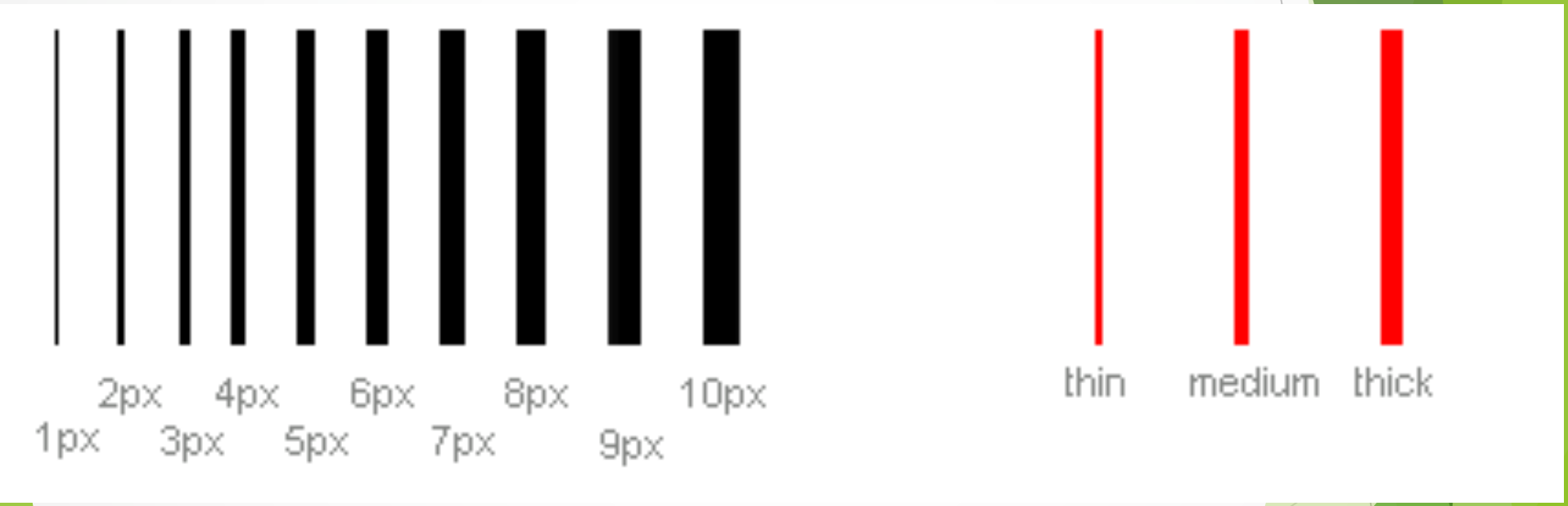
- border-width: thin, medium, thick or numerical value (e.g. 10px)
- border-color: color alias or RGB value
- border-style: none, hidden, dotted, dashed, solid, double, groove, ridge, inset, outset

Each property can be defined separately for left, top, bottom and right

- border-top-style, border-left-color, …
- border-radius (rounded the corner): border-radius (shorthand property), border-top-left-radius, border-top-right-radius, border-bottom-right-radius, border-bottom-left-radius

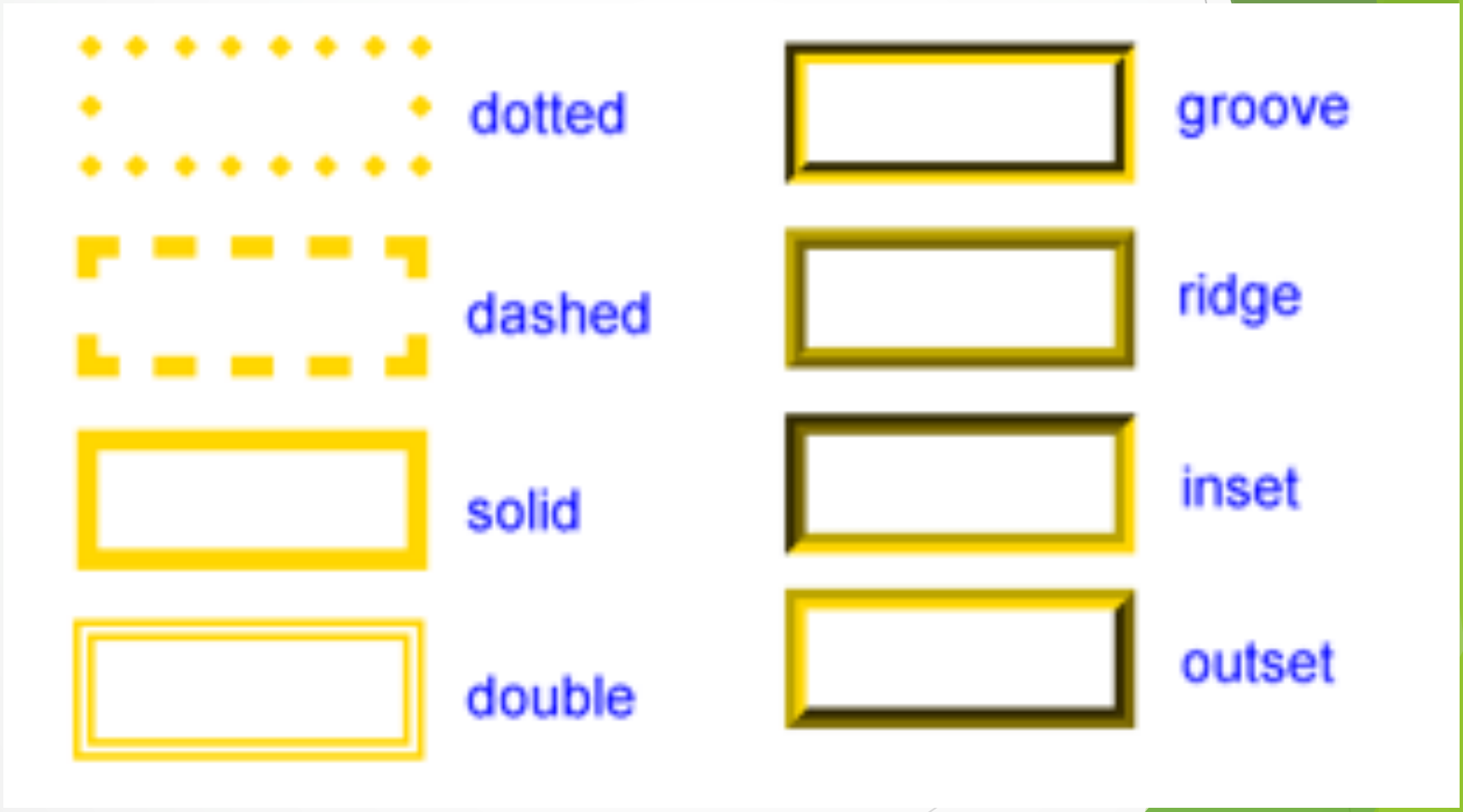
CSS Border (cont.)

- border-width : [value]



CSS Border (cont.)

- border-style



CSS Border Shorthand

- border: shorthand rule for setting border properties at once:

```
border: 1px solid red;
```

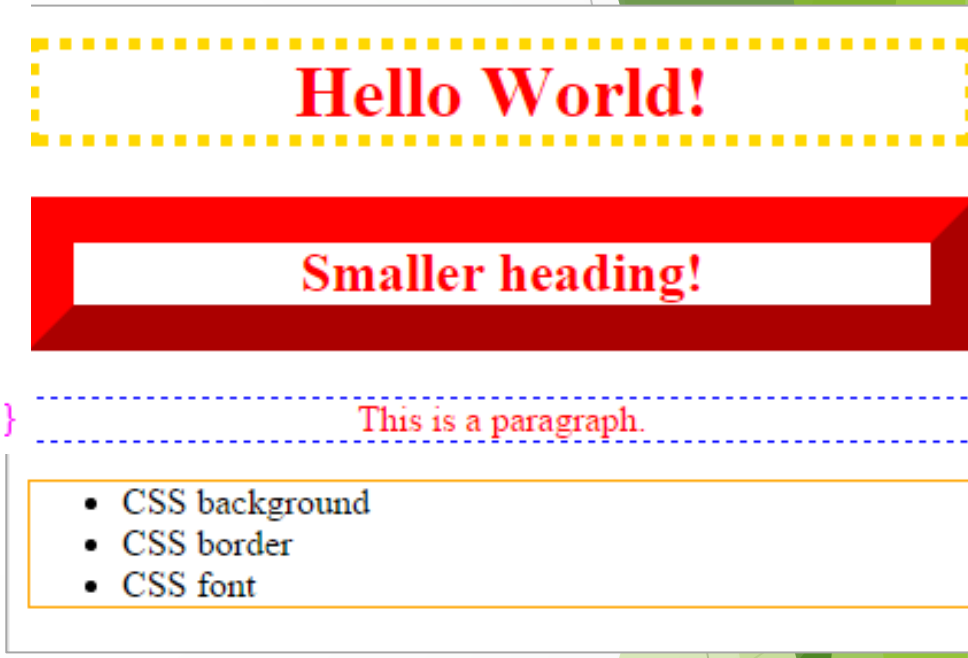
is equal to writing:

```
border-width:1px;
border-style:solid;
border-color:red;
```

- Specify different borders for the sides via shorthand rules: border-top, border-left, border-right, border-bottom
- When to avoid border:0

CSS Border Example

```
h1 {  border-width: thick;
      border-style: dotted;
      border-color: gold; }
h2 {   border-width: 20px;
      border-style: outset;
      border-color: red; }
 p {    border-width: 1px;
      border-style: dashed;
      border-color: blue; }
 ul { border: thin solid orange; }
```



CSS Border radius

One value: 15px:

Two values: 15px 50px:

Three values: 15px 50px 30px:

Four values: 15px 50px 30px 5px:

- One value: all four corners are rounded equally
- Two values:
  - First-value: top-left and bottom-right
  - Second-value: top-right and bottom-left
- Three values:
  - First-value: top-left
  - Second-value: top-right and bottom-left
  - Third value: bottom-right
- Four values:
  - First value: top-left
  - Second-value: top-right
  - Third-value: bottom-right
  - Fourth-value: bottom-left

CSS Width and Height

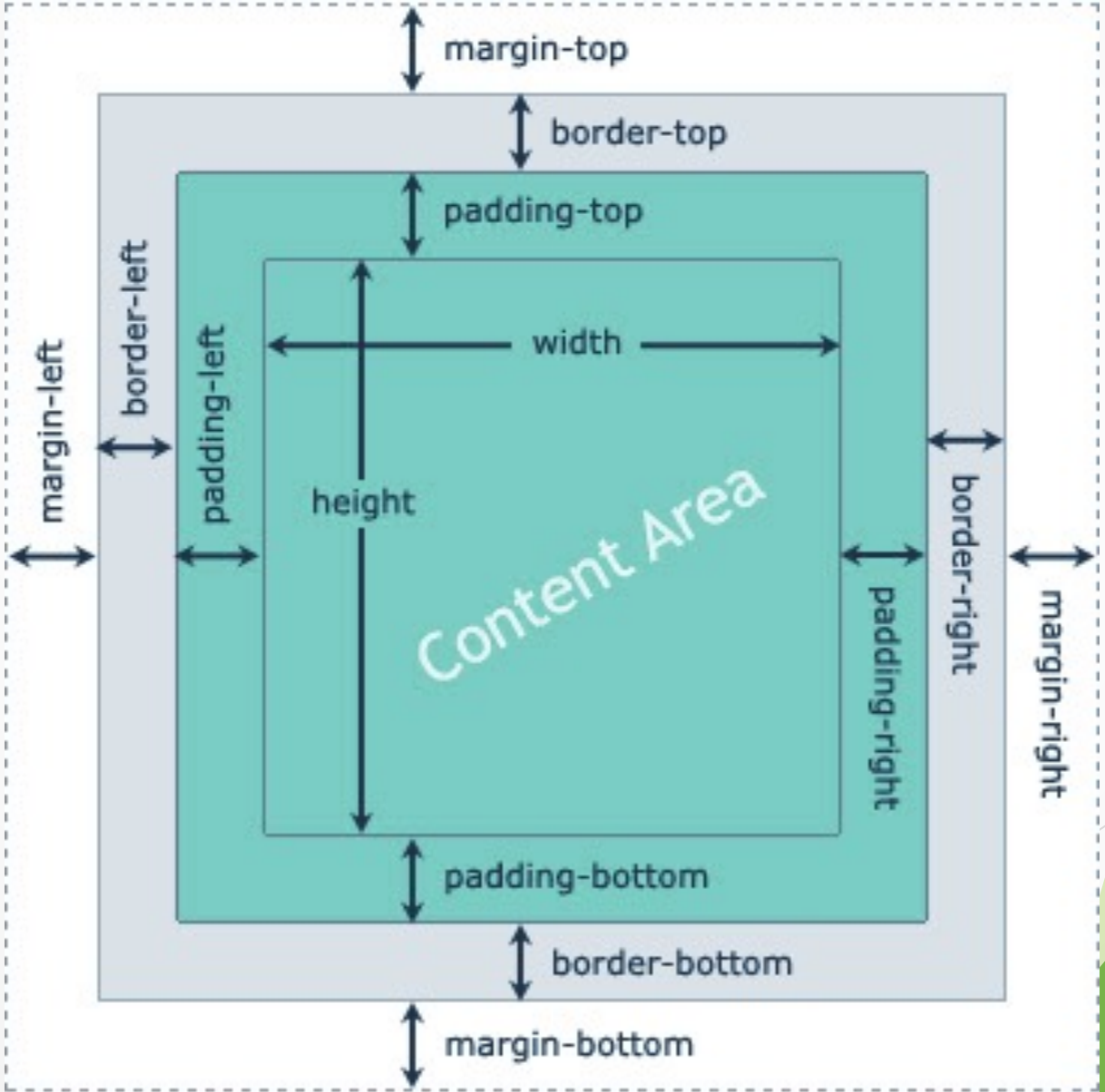
- width – defines numerical value for the width of element, e.g. 200px
- height – defines numerical value for the height of element, e.g. 100px
  - By default the height of an element is defined by its content
  - Inline elements do not apply height, unless you change their display style.
- Max-width: (not change when resize the screen size)

```
div {
    width: 500px;
    height: 100px;
    border: 3px solid #73AD21;
}
```

Set height and width of an Element:

This div element has a height of 100px and a width of 500px.

CSS Margin and Padding



CSS Margin and Padding

- margin and padding define the spacing around the element
  - Numerical value, e.g. 10px or -5px
  - Can be defined for each of the four sides separately - margin-top, padding-left, …
  - margin is the spacing outside of the border
  - padding is the spacing between the border and the content
  - What are collapsing margins?

Margin and Padding: Short Rules

- margin: 5px;
  - Sets all four sides to have margin of 5 px;
- margin: 10px 20px;
  - top and bottom to 10px, left and right to 20px;
- margin: 5px 3px 8px;
  - top 5px, left/right 3px, bottom 8px
- margin: 1px 3px 5px 7px;
  - top, right, bottom, left (clock wise from top)
- *Same for padding*

CSS Position

- `position`: defines the positioning of the element in the page content flow
- The value is one of:
  - `static` (default)
  - `relative` – relative position according to where the element would appear with static position
  - `absolute` – position according to the innermost positioned parent element
  - `fixed` – same as absolute, but ignores page scrolling

CSS Position (cont.)

- Margin VS relative positioning
- Fixed and absolutely positioned elements do not influence the page normal flow and usually stay on top of other elements
  - Their position and size is ignored when calculating the size of parent element or position of surrounding elements
  - Overlaid according to their z-index
  - Inline fixed or absolutely positioned elements can apply height like block-level elements

CSS Position (cont.)

- `top`, `left`, `bottom`, `right`: specifies offset of absolute/fixed/relative positioned element as numerical values
- `z-index` :
  - specifies the stack order of an element (which element should be placed in front of, or behind, the others)
  - An element with greater stack order is always in front of an element with a lower stack order
  - An element can have a positive or negative stack order
  - This property only applies for positioned elements (position: absolute/ fixed).

CSS Position (cont.)

```
<body>
<div style="background-color:lightblue; width:300px;
height:100px; position:relative; top:10px; left:80px; z-
index:2"></div>
<div style="background-color:yellow; width:300px;
height:100px; position:relative; top:-60px; left:35px; z-
index:1"></div>
<div style="background-color:lightgreen; width:300px;
height:100px; position:relative; top:-220px; left:120px; z-
index:3"></div>
</body>
```

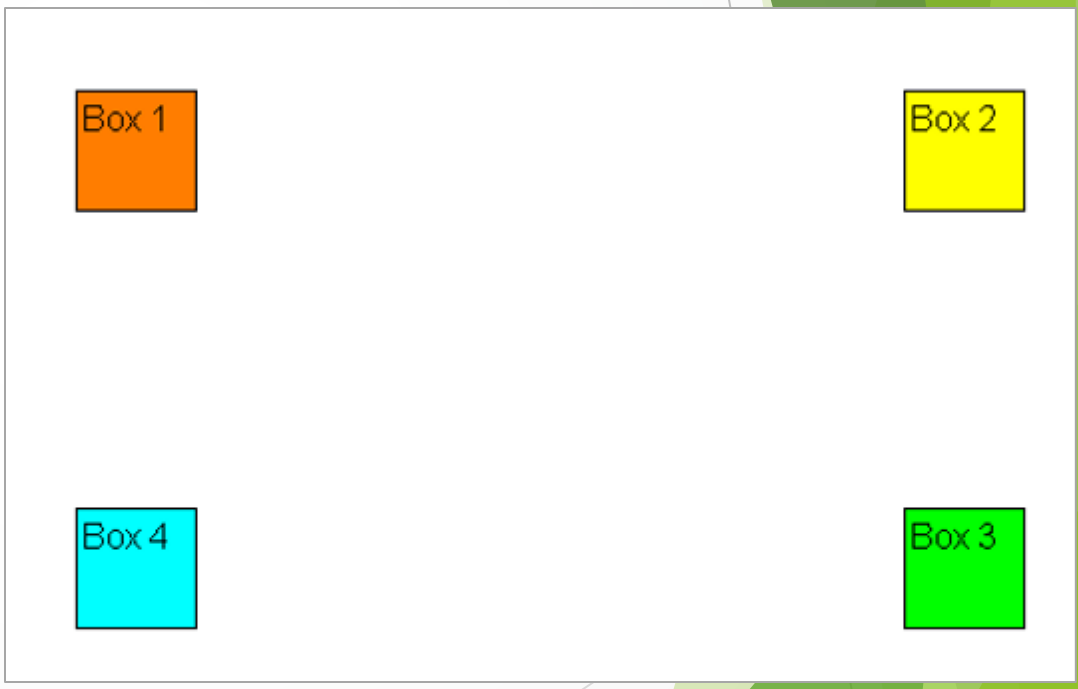
This is a heading

Because the image has a z-index of -1, it will be placed behind the text.

```
img {
    position: absolute;
    left: 0px;
    top: 0px;
    z-index: -1;
}
```

CSS Position (cont.)

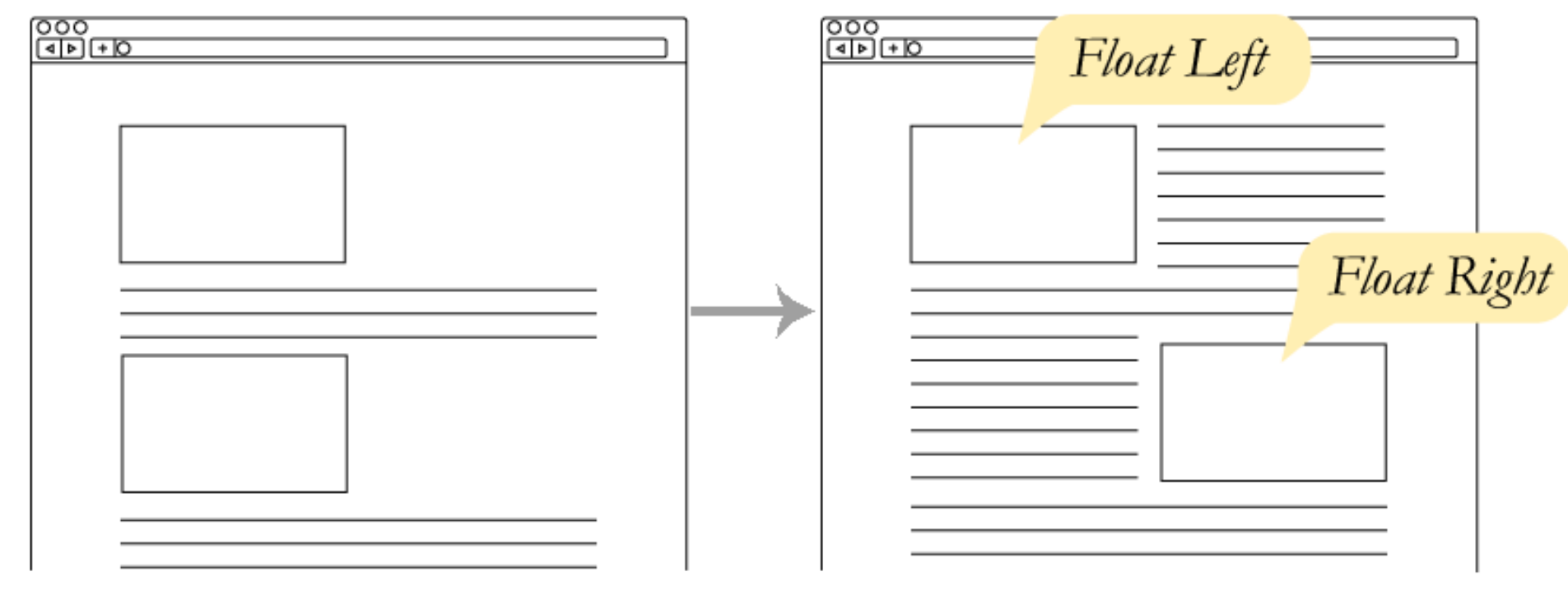
```
#box1 {
    position:absolute;
    top: 50px;
    left: 50px; }
#box2 {
    position:absolute;
    top: 50px;
    right: 50px; }
#box3 {
    position:absolute;
    bottom: 50px;
    right: 50px; }
#box4 {
    position:absolute;
    bottom: 50px;
    left: 50px; }
```



CSS Float

- float: the element “floats” to one side
  - left: places the element on the left and following content on the right
  - right: places the element on the right and following content on the left
  - none: the element does not float (will be displayed just where it occurs in the text). This is default
  - initial: default value
  - inherit: the element inherits the float value of its parent

CSS Float (cont.)



- Floating: normal
- Floating: left, right

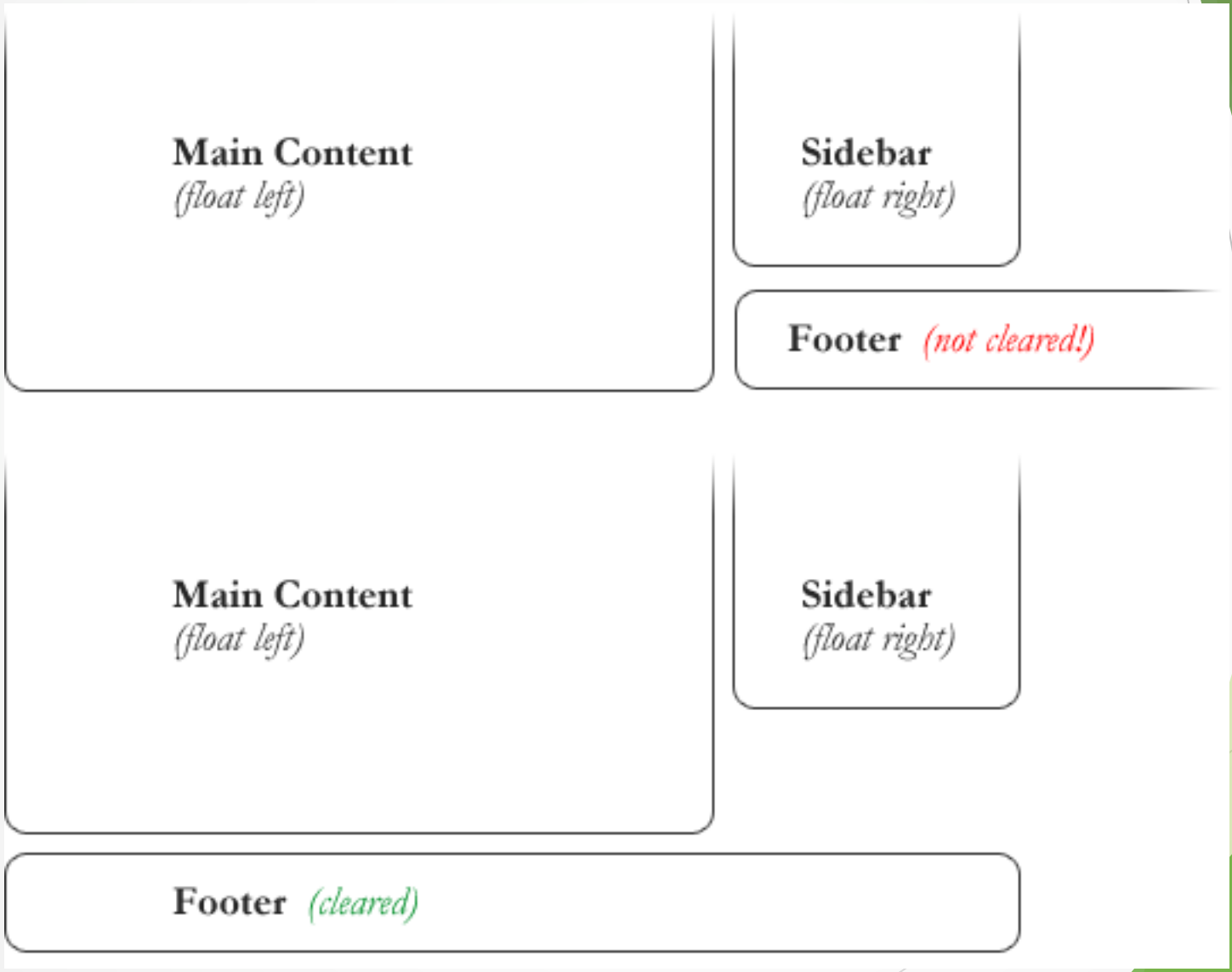
CSS Clear

- clear
  - Sets the sides of the element where other floating elements are NOT allowed
  - Used to "drop" elements below floated ones or expand a container, which contains only floated children
  - Possible values: left, right, both, initial, inherit
- Clearing floats
  - additional element ( ) with a clear style

CSS Clear (cont.)

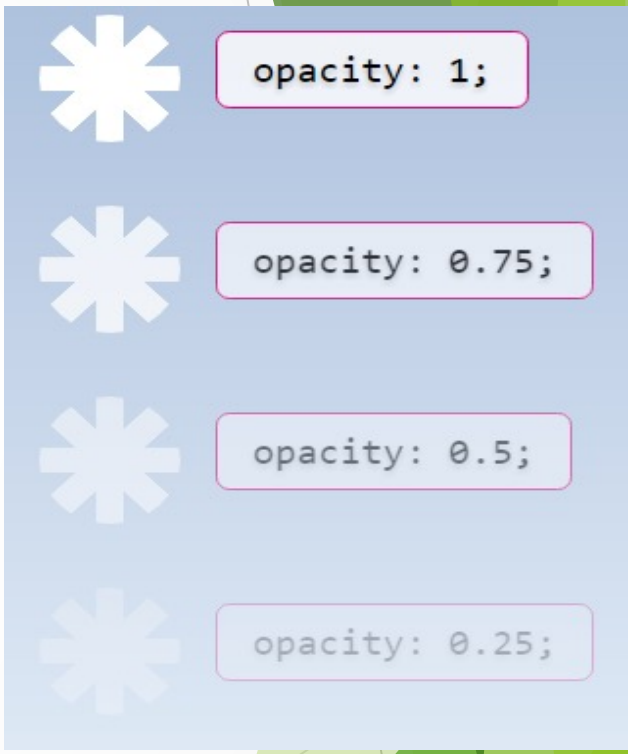
- Clearing floats (continued)
  - :after { content: ""; display: block; clear: both; height: 0; }
  - Triggering has Layout in IE expands a container of floated elements
    - display: inline-block;
    - zoom: 1;

CSS Clear (cont.)



CSS Opacity

- opacity: specifies the opacity of the element
  - Floating point number from 0 to 1
  - For old Mozilla browsers use `-moz-opacity`
  - For IE use `filter:alpha(opacity=value)` where value is from 0 to 100; also, "binary and script behaviors" must be enabled and `hasLayout` must be triggered, e.g. with `zoom:1`

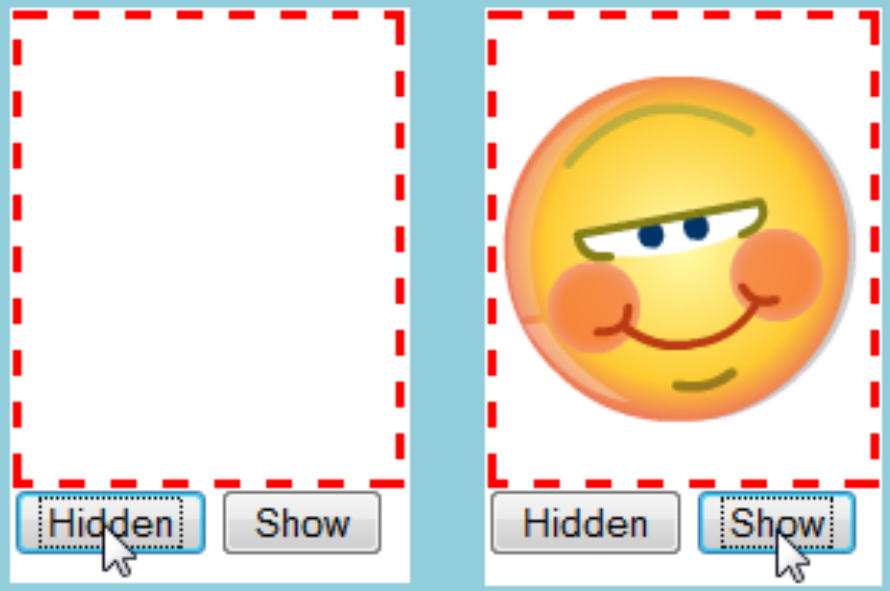


CSS Visibility

- visibility
  - Determines whether the element is visible
  - hidden: element is not rendered, but still occupies place on the page (similar to opacity:0)
  - visible: element is rendered normally
  - collapse: this is for use with dynamic table columns and rows effects.
  - initial: default value
  - Inherit: inherits this property from its parent element

CSS Visibility (cont.)

```
.frame
{
  border: dashed medium red;
  width:130px;
}

html file
<body>
<div class="frame">
<p><img id="myImg" src="smilec.png"></p>
</div>
<input type="button"
       value="Hidden"
       onclick="myImg.style.visibility='hidden'">

<input type="button"
       value="Show"
       onclick="myImg.style.visibility='visible'">
</body>
```



CSS Display

- display: is CSS's most important property for controlling layout. Specifies if/how an element is displayed.
  - none: the element is completely removed
  - inline: Displays an element as an inline element (like). Any height and width properties will have no effect
  - block: Displays an element as a block element (like <p>). It starts on a new line, and takes up the whole width
  - flex: Displays an element as a block-level flex container
  - grid: Displays an element as a block-level grid container

CSS Display (cont.)

- inline – block: Displays an element as an inline-level block container. The element itself is formatted as an inline element, but you can apply height and width values
- inline-flex: Displays an element as an inline-level flex container
- inline-grid: Displays an element as an inline-level grid container
- There are some more possible values, but not all browsers support them
  - Specific displays like table-cell and table-row

CSS Display (cont.)

```
<a href="/html/default.asp" target="_blank">HTML</a>
<a href="/css/default.asp" target="_blank">CSS</a>
<a href="/js/default.asp" target="_blank">JavaScript</a>
```

```
a {
  display: none;
}
```

```
a {
  display: inline;
}
```

HTML CSS JavaScript

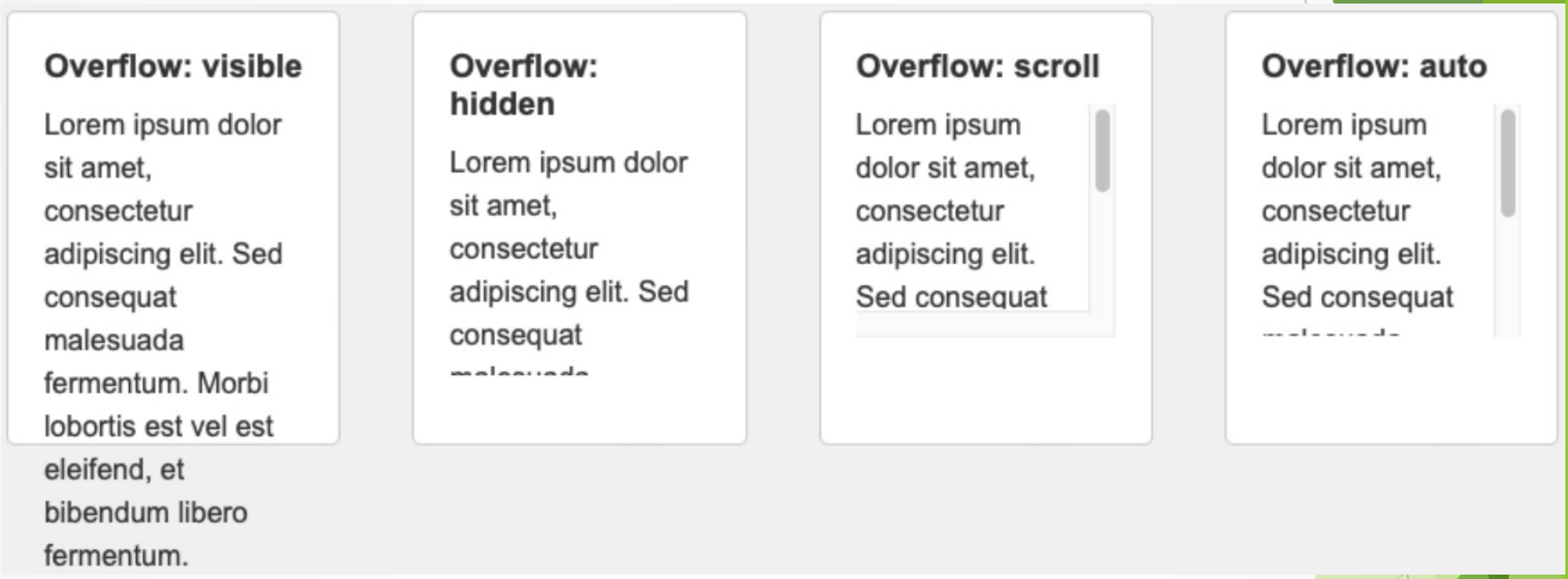
```
a {
  display: block;
}
```

HTML
CSS
JavaScript

Overflow

- overflow: defines the behavior of element when content needs more space than you have specified by the size properties or for other reasons. Values:
  - visible (default) – content spills out of the element
  - auto - show scrollbars if needed
  - scroll – always show scrollbars
  - hidden – any content that cannot fit is clipped
  - initial – default value
  - Inherit: inherits this property from its parent element

Overflow (cont.)



Center Align – Using margin

- Setting the width of block-level element will prevent it from stretching out to the edges of its container.
- Using margin:auto; to horizontally center an element with in its container -> the element will then take up the specified width and the remaining space will be split equally between the two margins

```
.center {
    margin: auto;
    width: 60%;
    border: 3px solid #8AC007;
    padding: 10px;
}
```

CSS Layout – width and max-width

- Problem:
  - When use: `width` and `margin:auto` → when the browser window is smaller then the width of the element. The browser will add a horizontal scrollbar to the page.
- Solution: using `max-width` instead → will improve the browser's handling of small windows.

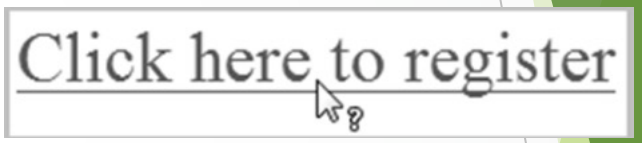
```
div.ex1 {
  width: 500px;
  margin: auto;
  border: 3px solid
#8AC007;
}
```

```
div.ex2 {
 max-width: 500px;
 margin: auto;
 border: 3px solid
#8AC007;
}
```

Screen Properties

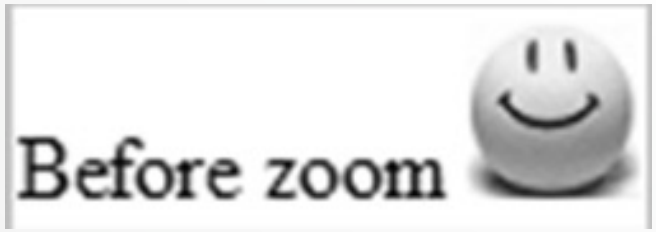
- The cursor property: specifies the type of cursor to be displayed when the mouse pointer is placed over the content.
- Values of the cursor property: default, hand, pointer, crosshair, text, wait, help, …

```
a { cursor: wait; }
```

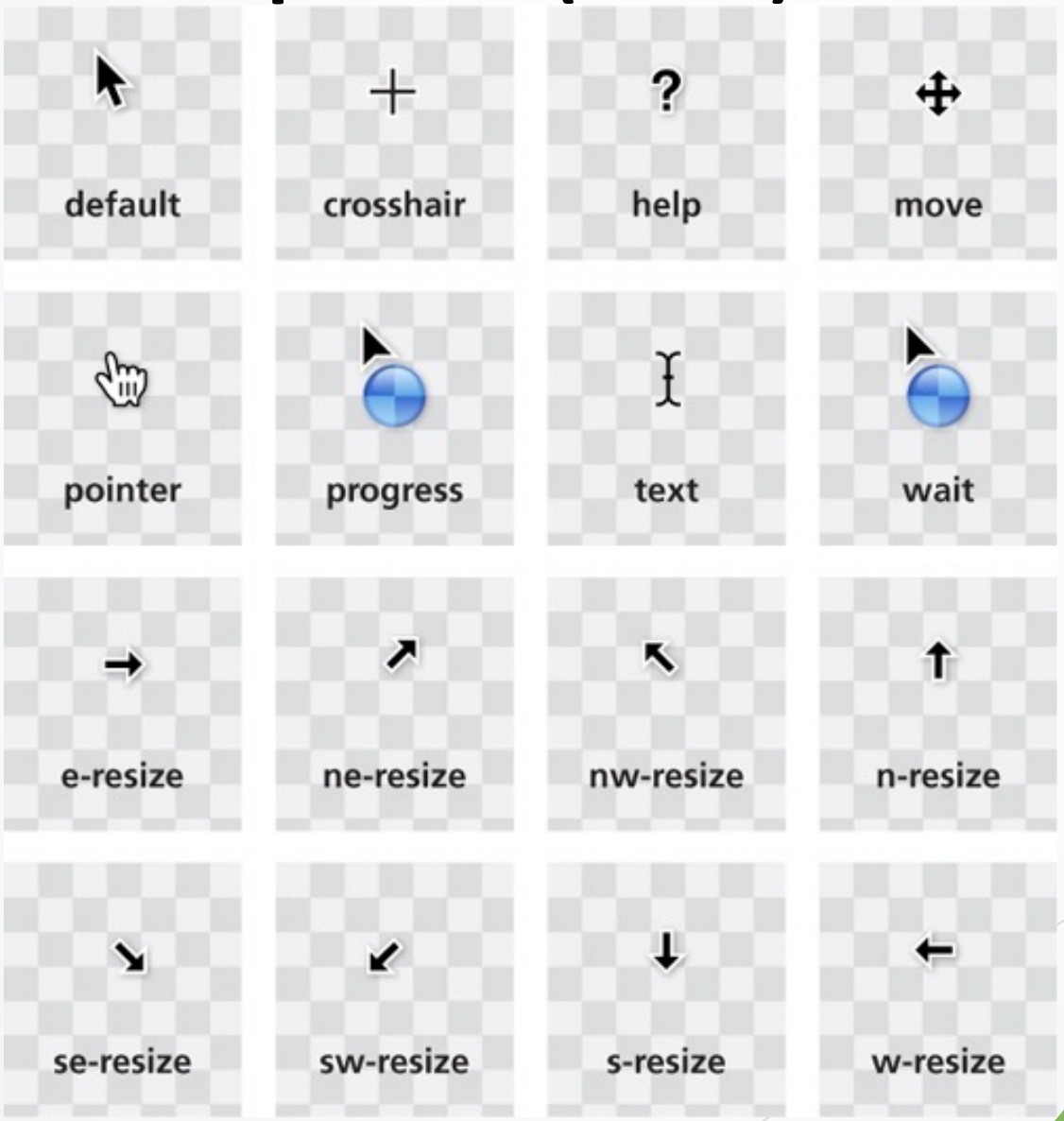


- The zoom property: enlarge an element on hover with CSS.

```
img { zoom: 200%; }
```



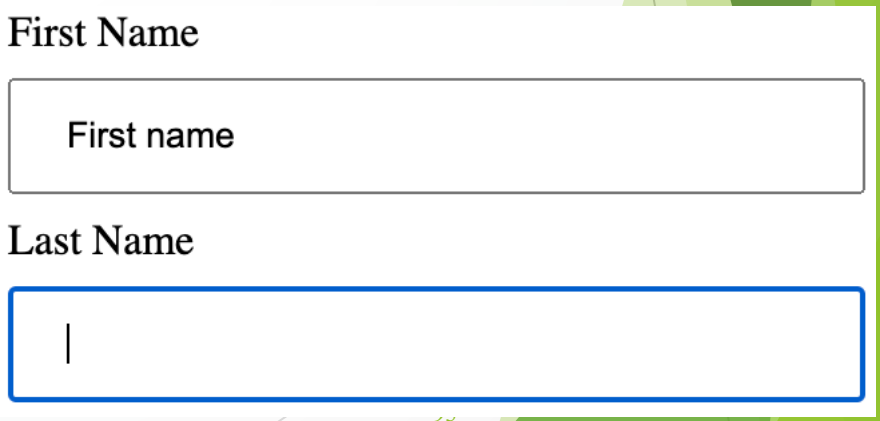
Screen Properties (cont.)



CSS Form

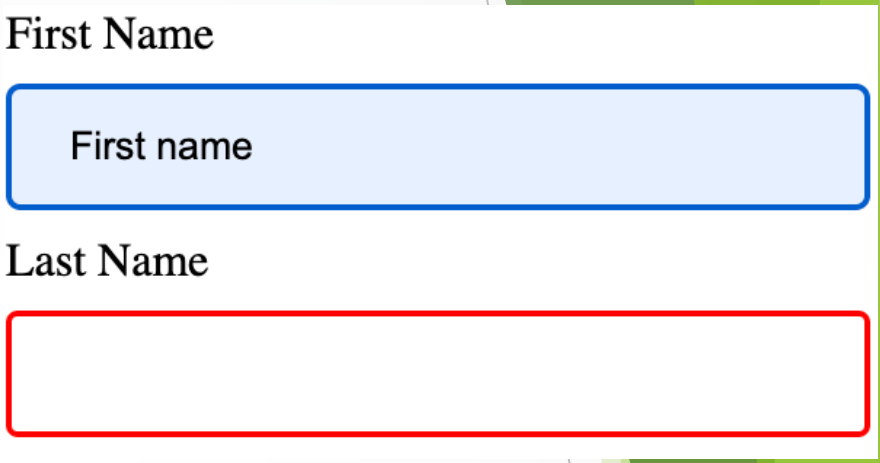
- Selector:
  - input: apply for all times
  - input[type=text]: only select text fields
  - input[type=password]: only select password fields
  - input[type=number]: only select number fields

```
input[type=text] {
  width: 100%;
  padding: 12px 20px;
  margin: 8px 0;
  box-sizing: border-box;

}
```



CSS Form (cont.)

```
input[type=text] {
  width: 100%;
  padding: 12px 20px;
  margin: 8px 0;
  box-sizing: border-box;
  border: 2px solid red;
  border-radius: 4px;
}

input[type=text]:focus {
  background-color: lightblue;
}

select {
  width: 100%;
  padding: 16px 20px;
  border: none;
  border-radius: 4px;
  background-color: #f1f1f1;
}
```



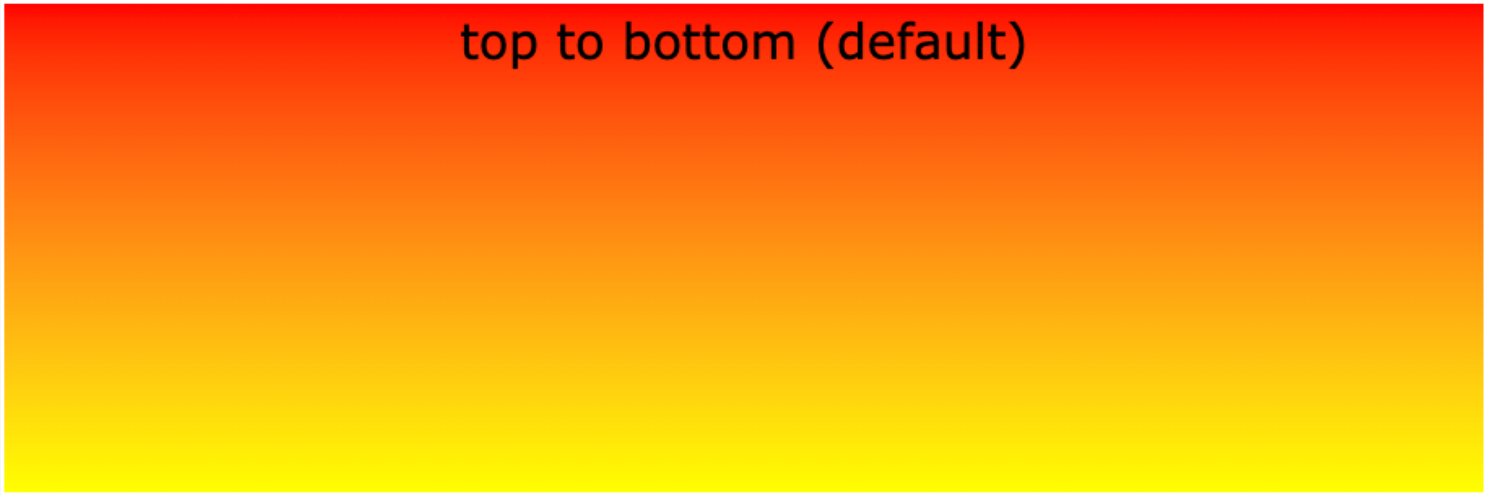
CSS Gradient

- CSS gradients let you display smooth transitions between two or more specified colors.
- CSS defines three types of gradients:
  - Linear Gradients (goes down/up/left/right/diagonally)
  - Radial Gradients (defined by their center)
  - Conic Gradients (rotated around a center point)

```
background-image: linear-gradient(direction, color-stop1, color-stop2, ...);
background-image: radial-gradient(shape size at position, start-color, ..., last-color);
background-image: conic-gradient([from angle] [at position,] color [degree], color [degree], ...);
```

CSS Gradient (cont.)

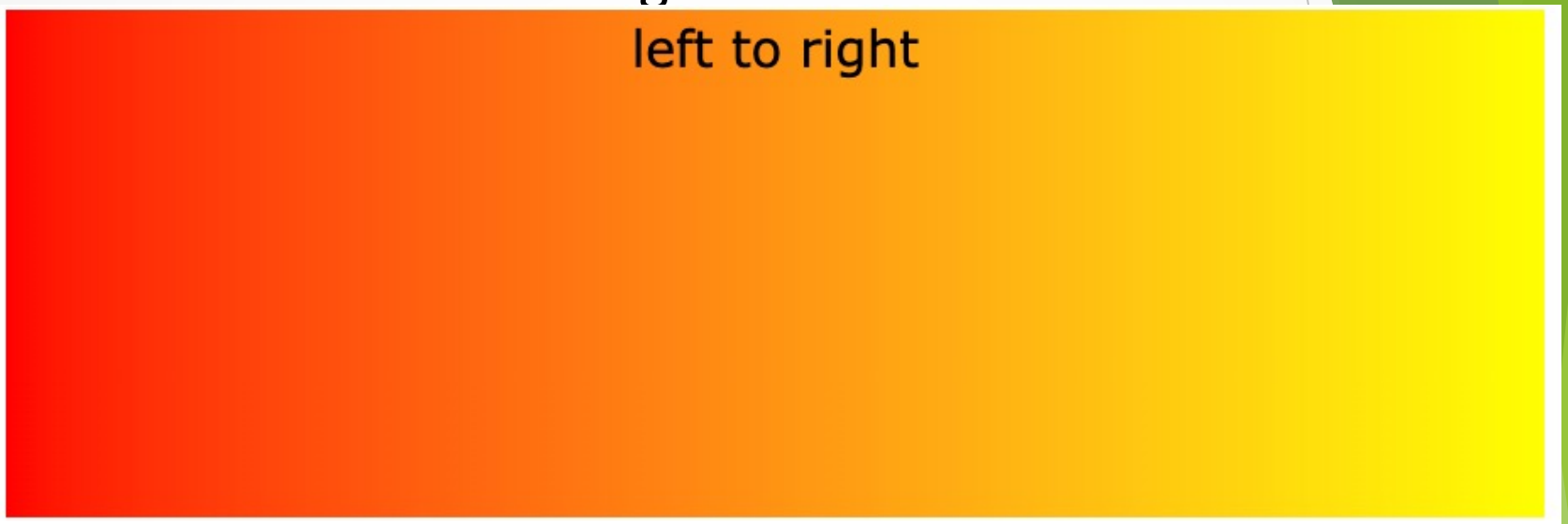
- Direction - Top to Bottom (this is default)




```
#grad {
  background-image: linear-gradient(red, yellow);
}
```

CSS Gradient (cont.)

- Direction – Left to Right




```
#grad {
  background-image: linear-gradient(to right, red
, yellow);
}
```

CSS Gradient (cont.)

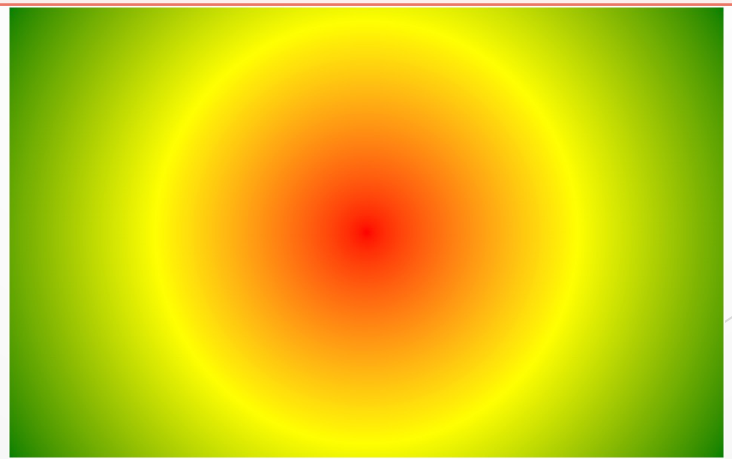
- Direction – Diagonal


```
#grad {
  background-image: linear-gradient(to bottom
right, red, yellow);
}
```

CSS Gradient (cont.)

- Radial gradient:
  - A radial gradient is defined by its center.
  - To create a radial gradient you must also define at least two color stops.

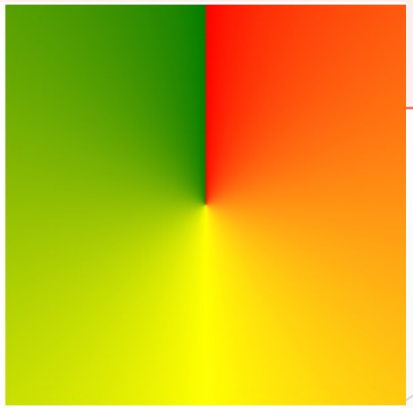
```
#grad {
  background-image: radial-gradient(circle, red, yellow, green);
}
```

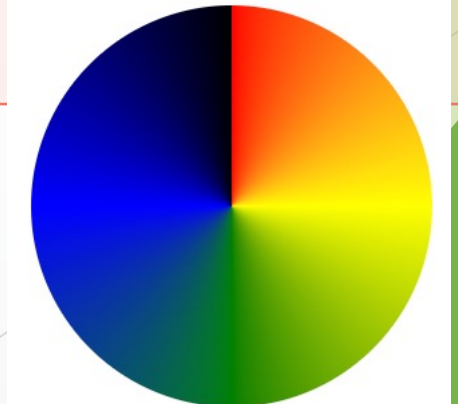


CSS Gradient (cont.)

- Conic gradient:
  - A conic gradient is a gradient with color transitions rotated around a center point.
  - To create a conic gradient you must define at least two colors.

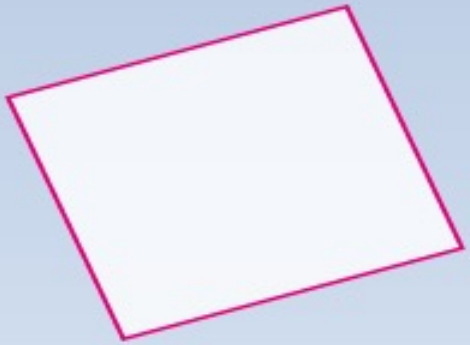
```
#grad {
  background-image: conic-gradient(red, yellow, green);
}
#grad {
  background-image: conic-gradient(red, yellow, green, blue, black);
  border-radius: 50%;
}
```





CSS Transform

- **translate**
```
transform: translate(-80px, 200px);
```
- **rotate**
```
transform: rotate(15deg);
```
- **scale**
```
transform: scale(1.5, 2);
```
- **skew**
```
transform: skewX(-8deg);
```



CSS Transitions

- CSS3 transition: allows to change property values smoothly (from one value to another), over a given duration
- To create a transition effect, must specify two things:
  - The CSS property you want to add an effect to
  - The duration of the effect
- Note: if the duration part is not specified, the transition will have no effect because the default value is 0

CSS Transitions (cont.)

- transition: shorthand property
- transition-delay: a delay (in seconds) for the transition effect
- transition-duration: how many seconds or milliseconds a transition effect takes to complete
- transition-property: name of the CSS property the transition effect
- transition-timing-function:
  - ease: slow start, then fast, hen end slowly (this is default
  - linear: same speed from start to end
  - ease-in: slow start ease-out-specifies a transition effect a slow end
  - ease-in-out: slow start and end

CSS Transitions (cont.)

```
div {
  width: 100px;
  height: 100px;
  background: red;
  transition: width 2s;
}
div:hover {
  width: 300px;
}
```

```
div {
  transition-property: width;
  transition-duration: 2s;
  transition-timing-function: linear;
  transition-delay: 1s;
}
div {
  transition: width 2s linear 1s;
}
```

CSS Animations

- CSS3 allows animation of HTML elements without using JavaScript or Flash
- An animation lest an element gradually change from one style to another.
- Can change as many CSS properties, as many times.
- Using @keyframes for the animation.
- Keyframes hold what styles the element will have at certain times.

CSS Animations (cont.)

```
/* The animation code */
@keyframes example {
  from {background-color: red;}
  to {background-color: yellow;}
}

/* The element to apply the animation to */
div {
  width: 100px;
  height: 100px;
  background-color: red;
  animation-name: example;
  animation-duration: 4s;
}
```

CSS Animations (cont.)

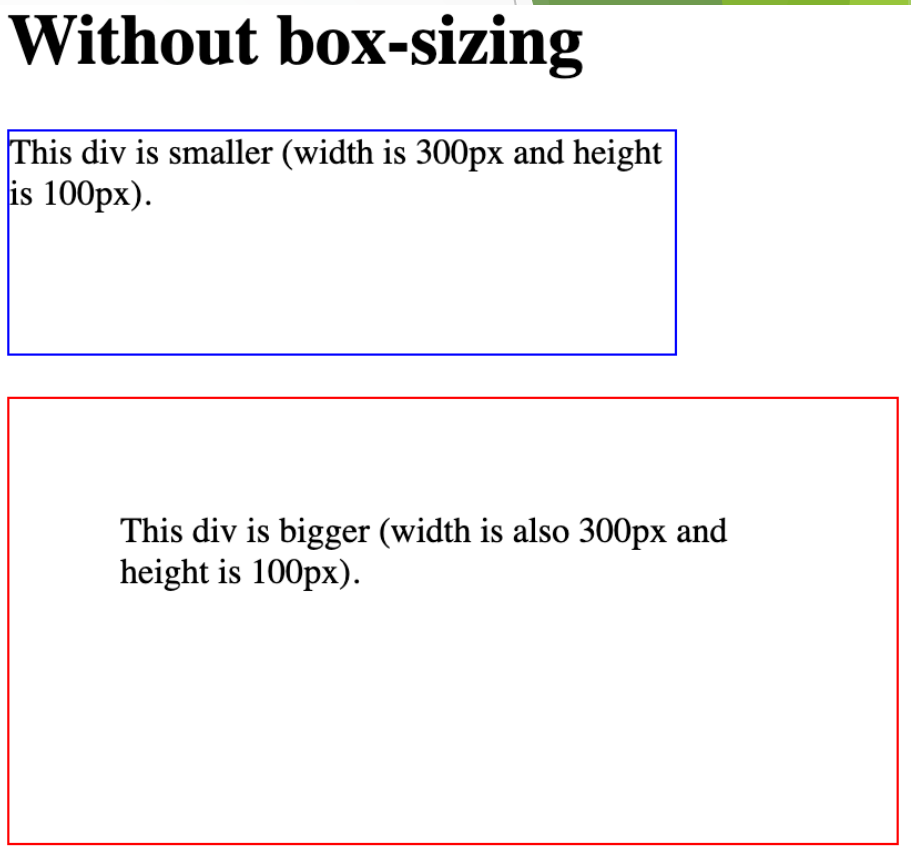
```
/* The animation code */
@keyframes example {
  0%   {background-color:red; left:0px; top:0px;}
  25%  {background-color:yellow; left:200px; top:0px;}
  50%  {background-color:blue; left:200px; top:200px;}
  75%  {background-color:green; left:0px; top:200px;}
  100% {background-color:red; left:0px; top:0px;}
}

/* The element to apply the animation to */
div {
  width: 100px;
  height: 100px;
  position: relative;
  background-color: red;
  animation-name: example;
  animation-duration: 4s;
}
```

CSS Box Sizing

- Without CSS3 box-sizing:
  - Actual width of an element = width + padding + border
  - Actual height of an element = height + padding + border
- With CSS3 box-sizing:
  - Include the padding and border in an element's total width and height.
  - Set `box-sizing: border_box;` on an element padding and border are included in the width and height

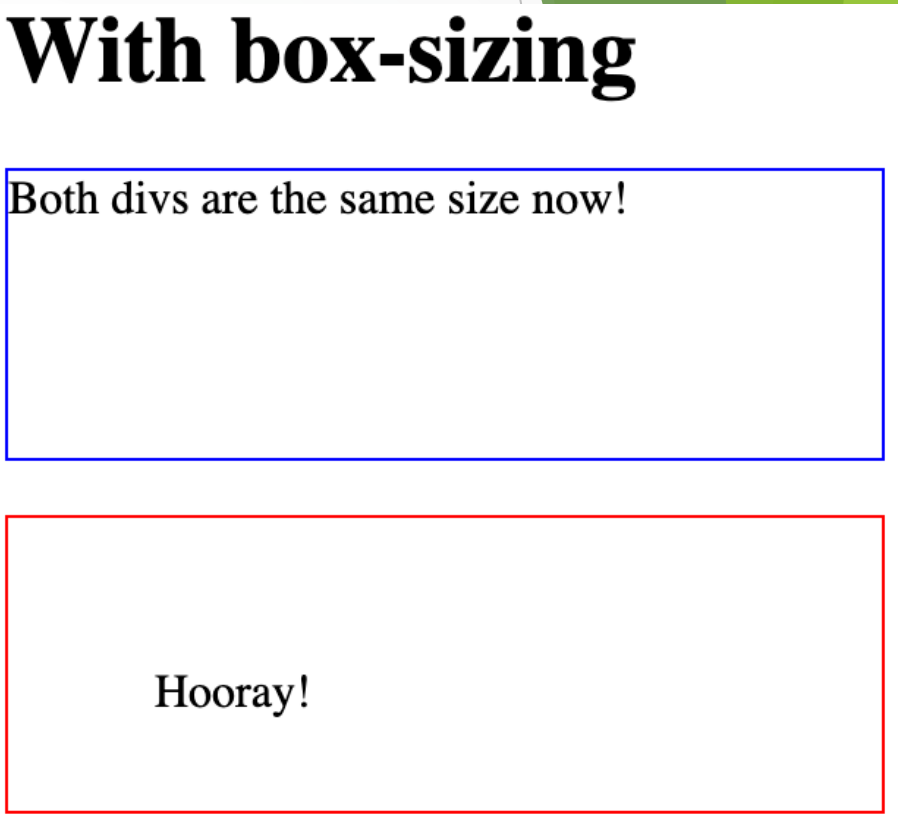
CSS Box Sizing (cont.)

```
.div1 {
   width: 300px;
   height: 100px;
   border: 1px solid blue;
}

.div2 {
   width: 300px;
   height: 100px;
   padding: 50px;
   border: 1px solid black;
}
```



CSS Box Sizing (cont.)

```
.div1 {
  width: 300px;
  height: 100px;
  border: 1px solid blue;
  box-sizing: border-box;
}

.div2 {
  width: 300px;
  height: 100px;
  padding: 50px;
  border: 1px solid red;
  box-sizing: border-box;
}
```



CSS Responsive Image

- Fixed width image → Problem

```
img {
  width: 200px; height: 300px;
}
```

- Solution:

```
img {
  width: 100%; height: auto;
}
```

- Even better:

```
img {
  max-width: 100%; height: auto;
}
```

CSS Multiple Columns

- The CSS multi-column layout allows easy definition of multiple columns of text – just like in newspapers
- Properties:
  - column-count: number of columns should be divided into
  - column-fill: how to fill columns
  - column-gap: the gap between the columns
  - column-rule: shorthand property for setting all the column-rule-* properties (color, style, width)
  - column-width
  - column-span: how many columns should span across
  - columns: shorthand property for setting column-width and column-count

CSS Multiple Columns (cont.)

```
.newspaper {
  column-count: 3;
  column-gap: 40px;
  column-rule: 1px solid lightblue;
}
h2 { column-span: all; }
```

Lorem Ipsum Dolor Sit Amet

Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat. Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero eros et accumsan et iusto odio dignissim qui blandit praesent luptatum zzril delenit augue duis dolore te feugait nulla facilisi. Nam liber tempor cum soluta nobis eleifend option congue nihil imperdiet doming id quod mazim placerat facer possim assum.

The @media rule

- The @media rule:
  - Allows to define different style rules for different media stypes
  - Example: set type rules for computer screens, printer, …
- Media queries can be used to check many things: as:
  - width and height of the viewport
  - width and height of the device
  - Orientation (is the table/phone in landscape or portrait mode)
  - resolution

The @media rule (cont.)

- Syntax:

```
@media
not|only mediatype and (mediafeature and|or|
not mediafeature) {
  CSS-Code;
}
```

- You can also have different stylesheets for different media, like this:

```
<link rel="stylesheet" media="mediatype
and|not|only
(mediafeature and|or|not mediafeature)
href"print.css"
```

CSS3 media type

- all: used for all media type devices
- print: used for printers
- screen: used for computer screens, tablets, smart-phone,…
- speech: used for screen readers that “readers” the page out loud

The @media rule (cont.)

```
/* Set the background color of body to tan */
body { background-color: tan;}

/* On screens that are 992px or less, set the
background color to blue */
@media screen and (max-width: 992px) {
  body { background-color: blue;}
}

/* On screens that are 600px or less, set the
background color to olive */
@media screen and (max-width: 600px) {
  body { background-color: olive; }
}
```

The @media rule (cont.)

```
@media screen and (max-width: 699px) and (min-
width: 520px), (min-width: 1151px){
   …
}

@media tv and (min-width: 700px) and
(orientation: landscape) {
 …
}

@media (min-width: 700px) and (orientation:
landscape) {
 …
}
```

Sitemap

- A sitemap is a list of pages of a web site within a domain. There are three primary kinds of sitemap:
  - Sitemaps used during the planning of a website by its designers
  - Human-visible listings typically hierarchical, of the pages on a site
  - Structured listings intended for web crawlers such as search engines

W3Schools Tutorials

HTML & CSS

» HTML Tutorial
» HTML5 Tutorial
» CSS Tutorial
» CSS3 Tutorial
» Bootstrap Tutorial

Browser Scripting

» JavaScript Tutorial
» jQuery Tutorial
» jQuery Mobile Tutorial
» AngularJS Tutorial
» AJAX Tutorial
» JSON Tutorial
» Google API Tutorial

XML Languages

» XML Tutorial
» DTD Tutorial
» XML DOM Tutorial
» XSLT Tutorial
» XPath Tutorial
» XQuery Tutorial
» Schema Tutorial
» RSS Tutorial
» XSL-FO Tutorial
» SVG Tutorial

Web Building

» Web Building
» Web Statistics
» Web Certification

Server Scripting

» PHP Tutorial
» SQL Tutorial
» ASP Tutorial
» VBScript Tutorial
» ASP.NET Tutorial
» ASP.NET WebPages Tutorial
» ASP.NET Razor Tutorial
» ASP.NET MVC Tutorial
» ASP.NET WebForms Tutorial
» Web Services Tutorial

Q&A