

# Chapter 4

## Cascading Style Sheets (CSS)

# CSS (cascading style sheets)

- A *style* is simply a rule describing how to format a particular portion of a web page. A *style sheet* is a set of these styles. i.e. A style sheet is a set of instructions that controls the appearance of a web page.
- separate language used to style documents
- uses rules to say how a document should appear
- CSS works with HTML, but it's not HTML.
- While HTML provides structure to a document by organizing information into headers, paragraphs, bulleted lists, and so on, CSS works hand-in-hand with the web browser to make HTML look good

# The Benefits of CSS

- Better type and layout controls.
- Less work. - change the appearance of an entire site by editing one style sheet.
- Potentially smaller documents and faster downloads.
- More accessible sites. - making it more accessible for mobile devices

# How the style sheet work

1. Start with a document that has been marked up in HTML.
2. Write style sheet rules for how you'd like certain element to look.
3. Attach the style rules to the document.

# Style Sheet Rules

- Style sheets are made up of one or more style instructions (called rules) for how a page element should be displayed.
- A rule says how a particular page element (whether it is heading, a paragraph, or block quote) should be displayed.
- There are two main sections of a rule those are:-
  - The selector :-which identifies the element to be affected(rule applies)
  - The declaration:-
    - The declaration is made up of a property and a value.
    - There can be more than one declaration in a single rule
    - Each declaration must end by semicolons.
    - Declarations are very similar to attribute names and their values

# Style Sheet Rules(cont...)

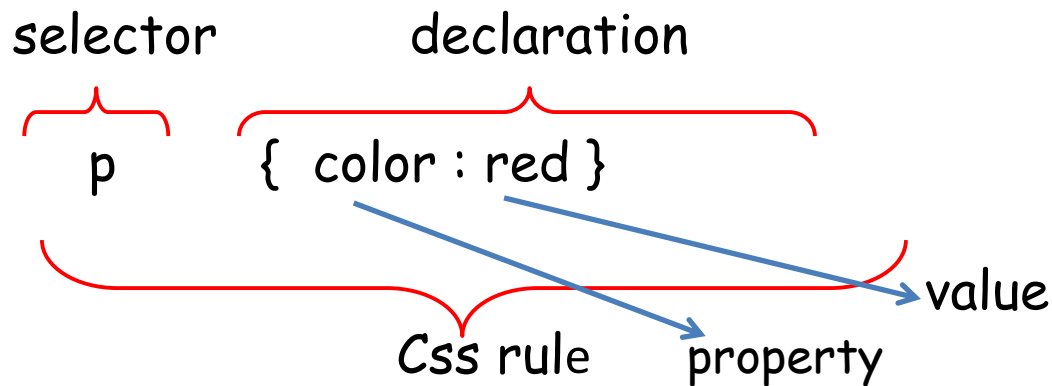
Syntax

Selector {property : value;}

or

```
Selector {property1 : value1;  
          property2: value2;  
          Property3:value3 ;  
          }
```

Example



# Attaching the styles to the document

- There are three way's that style information can be applied to an XHTML document.
  1. Inline style sheet
  2. Embedded style sheet
  3. External style sheet

# Inline style sheet

- Style information can be added to an individual element by adding the `STYLE` attribute within the `HTML` tag for that element.
- **Example1:** `<H1 STYLE = "color : red">`  
this heading will be red`</H1>`

**Will result :- this heading will be read**



# Embedded style sheet

- A more compact method for adding a style sheet is to embed a style block at the top of the HTML document using the <STYLE> tag and its rules apply only to that document.

Example:

```
<HTML>
<HEAD>
<STYLE TYPE="text/css">
H1 {color:red;}
P {font-size:12pt;
font-family:sans-serif;
}
</STYLE>
<TITLE>title</TITLE>
<h1>red heading </h1>
<p>paragraph 1</p>
</HEAD>
</HTML>
```

# Embedded style sheet

- Will result

**red heading**

paragraph 1

- Make sure that the `<STYLE>` tag contains the `TYPE="text/css"` attribute and it is placed within the `<HEAD>` of the document.

# External Style Sheets

- If you are applying one set of style instructions to a number of pages in a site, an external style sheet is the best method.
- It is a powerful because you can change the appearance of an entire site by making one change in the external style sheet document.
- First, put your style sheet rules in a separate document and save with .css suffix.
- Next, create a link to the style sheet document from within your web page (or pages) using a <LINK> tag.

# External Style Sheets

Example:

```
<HEAD>
```

```
< LINK REL="STYLE SHEET"
```

```
HREF="pathname/style.css" TYPE="text/css">
```

```
</HEAD>
```

- The REL attribute defines the linked document's relation to the document, that is a "Style sheet"

# Conflicting style

- Style information is passed down until it is overridden by style command with more weight
- The closer the style sheet is the content the more weight it is given
- To prevent a specific rule from being overridden you can assign it "important" with the !important indicator

Example `p {color : blue !important}`

# CSS Selectors

# CSS Selectors

## selector type

- Element selector
  - Grouped selectors
  - Descendant selectors
  - Id selectors
  - Class selectors
  - **Single Element selector**
    - Most basic type of selectors is element type selector.
- ```
P {color: blue;}
```

# Grouped selectors

- `p, ul, td, th {color :navy;}`
  - The disadvantage of selecting element this way of course, is that the property will apply to every paragraph and other listed elements in the document.



# Descendant selectors

- A Descendant selectors targets elements that contented within (therefore descendant of) another element.
- Descendant selectors are indicated in a list separated by a character space

## Example 1

```
li em {color: red}
```

- This example target emphasize text (em ) element that appear in list item (li)

# Descendant selectors (cont...)

- Example 2

`H1 em, h2 em, h3 em {color : red}`

This rule target em element that only when appear in h1 h2 and h3 heading

- It is also possible to nest descendant selector to nest several layer deep.

example

`ol a em {color : red}`

# ID selector

- We learned about the id attribute that gives an element and it's a unique identifying name
- The id attribute can be used with any HTML element, and it is commonly used to give meaning to the generic div and span elements
- The symbol that identifies ID selector is the # (hash) symbol

# ID selector

- Example

```
<li id="list1">computer science</li>
```

rule

```
#list1 {color : red}
```

Or

```
Li# list1 {color : red}
```

# Class selector

- Class name are indicated with a period (.)

Example

```
p.special {color : orange}
```

- To Apply a property to all elements of the same class, omit the element name in the selector

```
.special{color:red}
```

# Specificity

- refers to the fact that more specific selectors have more weight when it comes to handling style rule conflicts.
  - **ID selectors** are more specific than (and will override)
  - **Class selectors**, which are more specific than (and will override)
  - **Contextual selectors (Descendant selectors)**, which are more specific than (and will override)
  - **Individual element selectors**

eg. `<blockquote> <p id="par1">paragraph1</p> </blockquote>`

`p { line-height: 1.2em; }` - **Individual element selectors**

`blockquote p { line-height: 1em; }` - **Contextual selectors**

`p.intro { line-height: 2em; }` - **Class selectors**

`# par1 {line-height:2.5 em}`- **ID selectors**

Therefore 2.5em will apply to the paragraph

❖ **ID selectors > Class selectors > Contextual selectors > Individual element selectors**

# Pseudo class Selectors

- Pseudo selectors are indicated by the colon (:) character.
- **Anchor pseudo classes**
- There are four main pseudoclasses that can be used as selectors:
  - a:link :-Applies a style to unclicked (unvisited) links
  - a:visited :-Applies a style to links that have already been clicked
  - a:hover:- Applies a style when the mouse pointer is over the link
  - a:active :-Applies a style while the mouse button is pressed
- to use all four anchor pseudo classes in a single style sheet, they need to appear in a particular order in order to function properly. the required order is
  - :link, :visited, :hover, :active.

# Style Properties



# Text properties

- allow you to control the appearance of the text. it is possible :-
  - To change the color of a text
  - Increase or decrease the space between character in a text
  - Decorate a text and more

# Text properties

**Color property:** - Set the color of the text

- *Values: color value (name or numeric)*
- *Default: depends on the browser*
- *Applies to: all elements*
- *Inherits: yes*

# Text properties

- **Direction property** :- Set the text direction
  - **Value** :ltr | rtl
  - **Default** :ltr
  - **Inherits**: yes
- **Letter-spacing property** :-increase or decrease the space between character
  - **Value** : normal | length
- **Text-align property** :-align the text in an element
  - **Value** : left | right | center

# Text properties

- **Text-decoration** :- adds decoration to text
  - **Values:** none | underline | overline | line-through | blink
  - **Default:** none
  - **Applies to:** all elements
  - **Inherits:** no, but since lines are drawn across child elements they may look like they are “decorated” too
- **The text-indent property**:- indent the first line of text by a specified amount
  - Values:** length measurement, percentage
  - Default:** 0
  - Applies to:** block-level elements and table cells
  - Inherits:** yes

# Text properties

- **Text-transform:** - control the letter in an element
  - **Value:** none | capitalize | uppercase | lowercase
- The final two text properties are used to insert space between
  - letters (letter-spacing) or
  - words (word-spacing) when the text is displayed.

# Text properties

- letter-spacing
  - Values: length measurement, normal
  - Default: normal
  - Applies to: all elements
  - Inherits: yes
- word-spacing
  - Values: length measurement, normal
  - Default: normal
  - Applies to: all elements
  - Inherits: yes

# The Font Properties

- The Font Properties allow you to change the font family, boldness, size and the style of the text.
- font-family property:- It is the prioritized list of font family names and/or generic family names for an element.
  - **Values can be**:- one or more font or generic font family names, separated by commas
  - **Default**: depends on the browser
  - **Applies to**: all elements
  - **Inherits**: yes

# The Font Properties

- five generic font families:
  - **Serif fonts** - Times, Georgia, and New Century Schoolbook
  - **Sans-serif fonts** - Helvetica, Geneva, Verdana, and Arial,
  - **Monospace fonts** - Courier, and Courier New
  - **Cursive fonts** - Zapf Chancery, and Comic Sans.
  - **Fantasy fonts** - Algerian, and **Cooper Black**



# The Font Properties

- font-family property

e.g.

```
body { font-family: Arial; }  
td { font-family: Courier, monospace; }  
p { font-family: "Trebuchet MS", Verdana, sans-serif; }
```

Use quotation  
if the font type  
has space

- "Why specify more than one font?"
- Browsers are limited to displaying fonts that are already installed on the user's machine.
- CSS allows you to provide a list of back-up fonts should your first choice not be available.
- When you specify a generic font family, the browser chooses an available font from that stylistic category.

# Font size

- Sets the size of a font
    - **Value can be** :-length unit, percentage, xx-small | x-small | small | medium | large | x-large | xx-large | smaller | larger
    - **Default:** medium
    - **Applies to:** all elements
    - **Inherits:** yes
- E.g `p{font-size=15em}`



No space needed

# Font weight (boldness)

- Set the weight of a font
  - Values: normal | bold | bolder | lighter | 100 | 200 | 300 | 400 | 500 | 600 | 700 | 800 | 900 |
  - Default: normal
  - Applies to: all elements
  - Inherits: yes

E.g. `b {font-weight: bold;}`

normal | **bold** | **bolder** | lighter

100 | 200 | 300 | 400 | 500

**600** | **700** | **800** | **900**

normal | **bold** | **bolder** | lighter

100 | 200 | 300 | 400 | 500

600 | **700** | **800** | **900**

# font-style property

- The font-style property affects the **posture** of the text,  
font-style
  - Values: **normal | italic | oblique**
  - Default: **normal**
  - Applies to: **all elements**
  - Inherits: **yes**

Example

```
p {font-style: normal;}
```

# Font Variant (Small Caps)

- Displays text in a small-caps font or normal font
  - Values: normal | small-caps
  - Default: normal
  - Applies to: all elements
  - Inherits: yes

# The shortcut font property

- CSS provided the shorthand font property that compiles all the font-related properties into one rule.

## Font

- Values:** font-style font-weight font-variant font-size/line-height font-family
- Default:** depends on default value for each property listed
- Applies to:** all elements
- Inherits:** yes

# The shortcut font property

Example :-

```
h1 {font-family: Verdana, Arial, sans-  
    serif; font-size: 30px; font-weight:  
    900; font-style: italic; font-variant:  
    small-caps;}
```

is the same as

```
h1 {font: italic 900 small-caps 30px  
    Verdana, Helvetica, Arial, sans-serif;}
```

# foreground color

- You specify a foreground color with the color property

color

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



# Background properties

- Define the background effect of an element
- Background properties allow you to control the background color of an element, set an image as the background ,repeat the background image vertically or horizontally and position an image on a page.

# Background properties :-

- **Background-color property**:-set the background color
  - **value**:-color-name | color hex
- **Background Images property**
  - Adding a background image
  - is used to add a background image to an element.
    - Values**: URL (location of image) | **none**
    - Default**: **none**
    - Applies to**: all elements
    - Inherits**: no

# Background position

- Set the starting position of the background image
- background-position
  - **Values:** length measurement | percentage | left | center | right | top | bottom | inherit
  - **Default:** 0% 0% (same as left top)
  - **Applies to:** all elements
  - **Inherits:** no

# background-repeat

- Set if how a background image will be repeated
- background-repeat
  - **Values:** repeat | repeat-x | repeat-y | no-repeat | inherit
  - **Default:** repeat
  - **Applies to:** all elements
  - **Inherits:** no
- If you want a background image to appear just once, use the **no-repeat** keyword value

# Example

```
<html>
<head>
<style>
body{
background-image:URL(2.jpg);
background-repeat:no-repeat;
background-position:center;
}
</style>
</head>
<body>
<h1>welcome</h1>
</body>
</html>
```

# Background property

- **Background property** :-a shorthand property for setting all background properties in one declaration
  - **Value**:- background-color| background-image | background-position
  - **Default**: see individual properties
  - **Applies to**: all elements
  - **Inherits**: no

# Background property

## Example

```
body { background: black url(arlo.jpg) no-repeat right top fixed; }
```

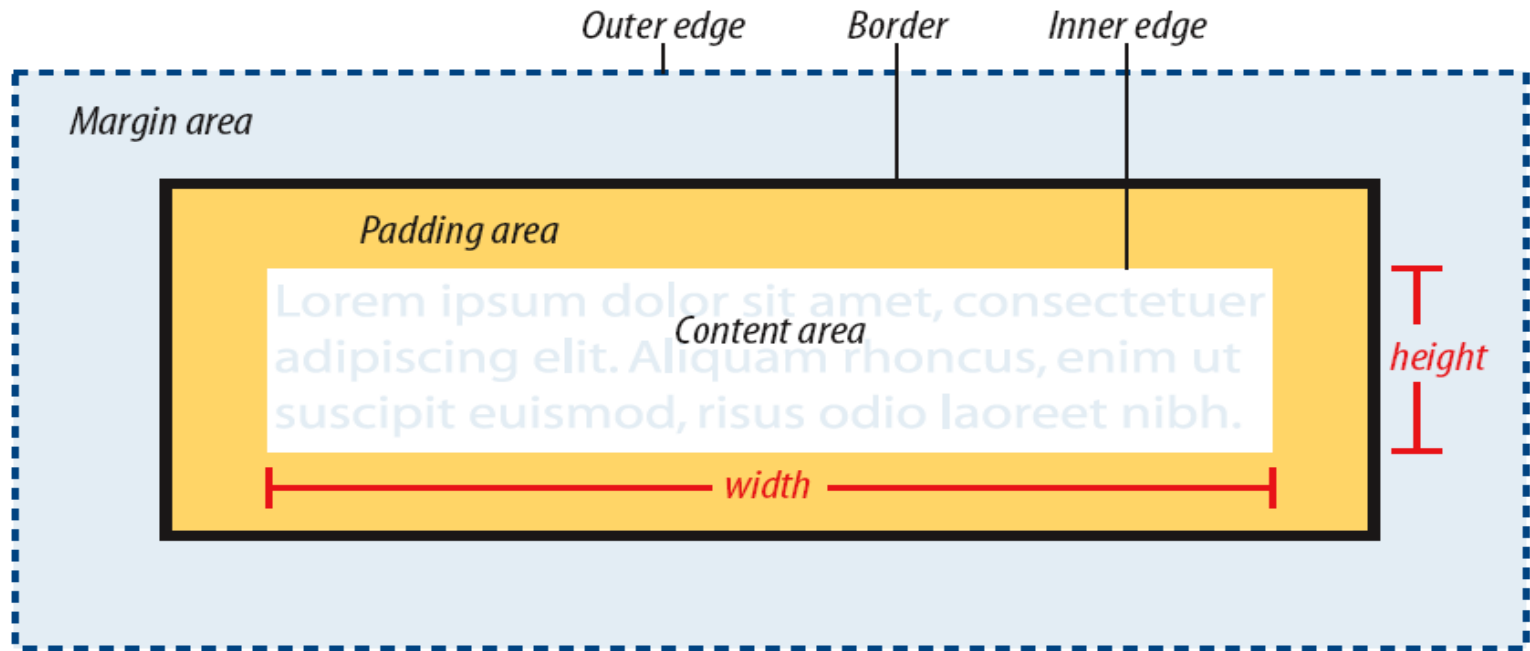
Or

replaces this rule with five separate declarations:

```
body {  
background-color: black;  
background-image: url(arlo.jpg);  
background-repeat: no-repeat;  
background-position: right top;}
```

# Padding, Borders, and Margins

- The browser sees every element on the page (both block and inline) as being contained in a little rectangular box.



**Figure 14-1.** The parts of an element box according to the CSS box model.



# Setting the Content Dimensions

- Use the width and height property to specify the width and height of the content area.
- **width**
  - **Values:** length measurement | percentage | auto
  - **Default:** auto
  - **Applies to:** Block-level elements and non-text inline elements (such as images)
  - **Inherits:** no
- **height**
  - **Values:** length measurement | percentage | auto
  - **Default:** auto
  - **Applies to:** Block-level elements and non-text inline elements (such as images)
  - **Inherits:** no

# Handling overflow

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

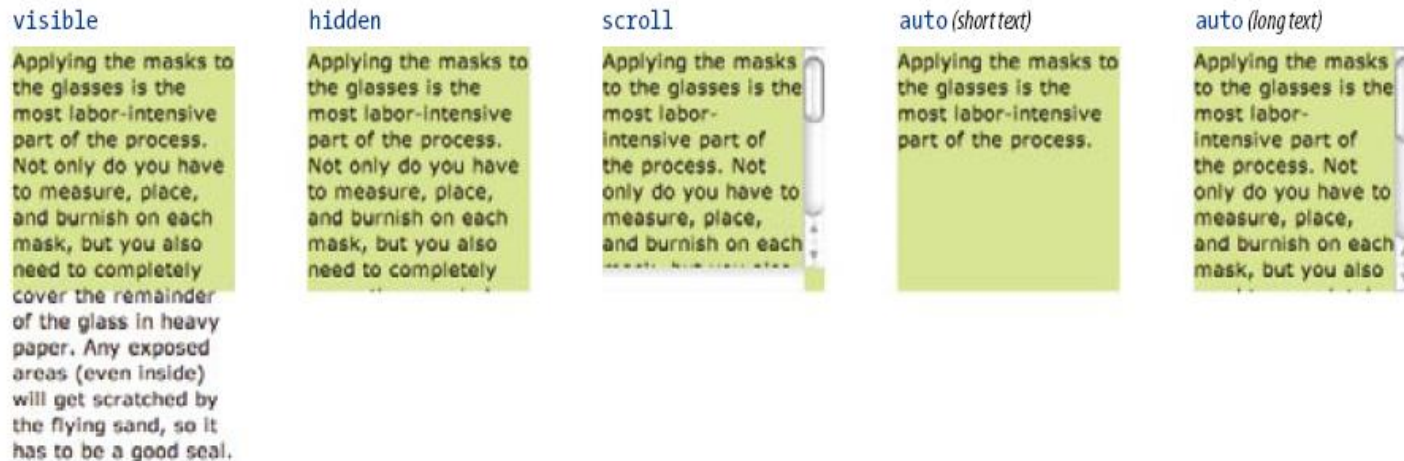


Figure 14-4. Options for handling content overflow.

# Border properties

- The border properties allow you to specify the style, color and width of the element border

# Border properties

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

# Border properties

- **border-top-style, border-right-style, border-bottom-style, border-left-style**
  - Values: none | dotted | dashed | solid | double | groove | ridge | inset | outset
  - Default: none
  - Applies to: all elements
  - Inherits: no

# Border properties

- **border-width**
  - Values: length units | thin | medium | thick | inherit
  - Default: medium
  - Applies to: all elements
  - Inherits: no

# Border properties

- `border-top-width`, `border-right-width`, `border-bottom-width`, `border-left-width`
  - Values: length units | thin | medium | thick | inherit
  - Default: medium
  - Applies to: all elements
  - Inherits: no

# Border properties

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



# Border properties

- **border-top-color, border-right-color, border-bottom-color, border-left-color**
  - Values: color name or RGB value | transparent | inherit
  - Default: the value of the color property for the element
  - Applies to: all elements
  - Inherits: no

# Border properties

- **Border property** :-a shorthand property for setting all of the property for the four border in one declaration .
  - Values : border-width |border-style |border- color

# Margins properties

- Define the space around element
- **margin-top, margin-right, margin-bottom, margin-left**
  - Values: length measurement | percentage | auto
  - Default: auto
  - Applies to: all elements
  - Inherits: no

# Margins properties

- **Margin property** :-a shorthand property for setting the margin properties in one declaration
  - Values: length measurement | percentage | auto | inherit
  - Default: auto
  - Applies to: all elements
  - Inherits: no

# Padding property

- padding property define the space between the element border and the element content.
- Padding is the space between the content area and the border
- **padding-top, padding-right, padding-bottom, padding-left**
  - Values: length measurement | percentage | auto | inherit
  - Default: auto
  - Applies to: all elements
  - Inherits: no

# Padding property

Padding:- a shorthand property for setting all of the padding properties in one declaration.

- Values: length measurement | percentage | auto | inherit
- Default: auto
- Applies to: all elements
- Inherits: no

# FLOATING AND POSITIONING

- **the float property** moves an element as far as possible to the left or right, allowing the following content to wrap around it.
  - Values: left | right | none | inherit
  - Default: none
  - Applies to: all elements
  - Inherits: no

# Positioning

- Is the way to specify the location of an element anywhere on the page with pixel precision
  - Values: static | relative | absolute | fixed
  - Default: static
  - Applies to: all elements
  - Inherits: no



# Positioning

- **Static** - elements are positioned as they occur in the normal document flow.
- **Relative** - **Relative positioning** moves the box relative to its original position in the flow. The space the element would have occupied in the normal flow is preserved.
- **Absolute** - **Absolutely positioned** elements are removed from the document flow entirely and positioned relative to a containing element
- **Fixed**- the element stays in one position in the window even when the document scrolls.

# Specifying position

- the actual position is specified with four offset properties.
- **top, right, bottom, left**
  - Values: length measurement | percentage | auto
  - Default: auto
  - Applies to: Positioned elements (where position value is relative, absolute, or fixed)
  - Inherits: no

# Questions?