COLLAGE OF COMPUTER SCIENCE & ENGINEERING

UNIVERSITY OF JEDDAH

كلية علوم و هندسة الحاسب

جامعة جدة

CSS: Cascading Style Sheets

CCSW 321 (Web Development)

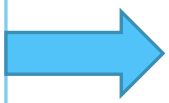
What will be covered

- Introduction.
- CSS Validation.
- CSS Selectors.
- CSS Properties.
- Inheritance.
- Conflicting Styles.
- Development Tips

What is CSS?

- CSS describes the appearance and layout of a web page.
- Used to specify the presentation of elements separately from the structure of the document.
- Composed of CSS rules, which define sets of styles

```
selector {
  property: value;
}
```



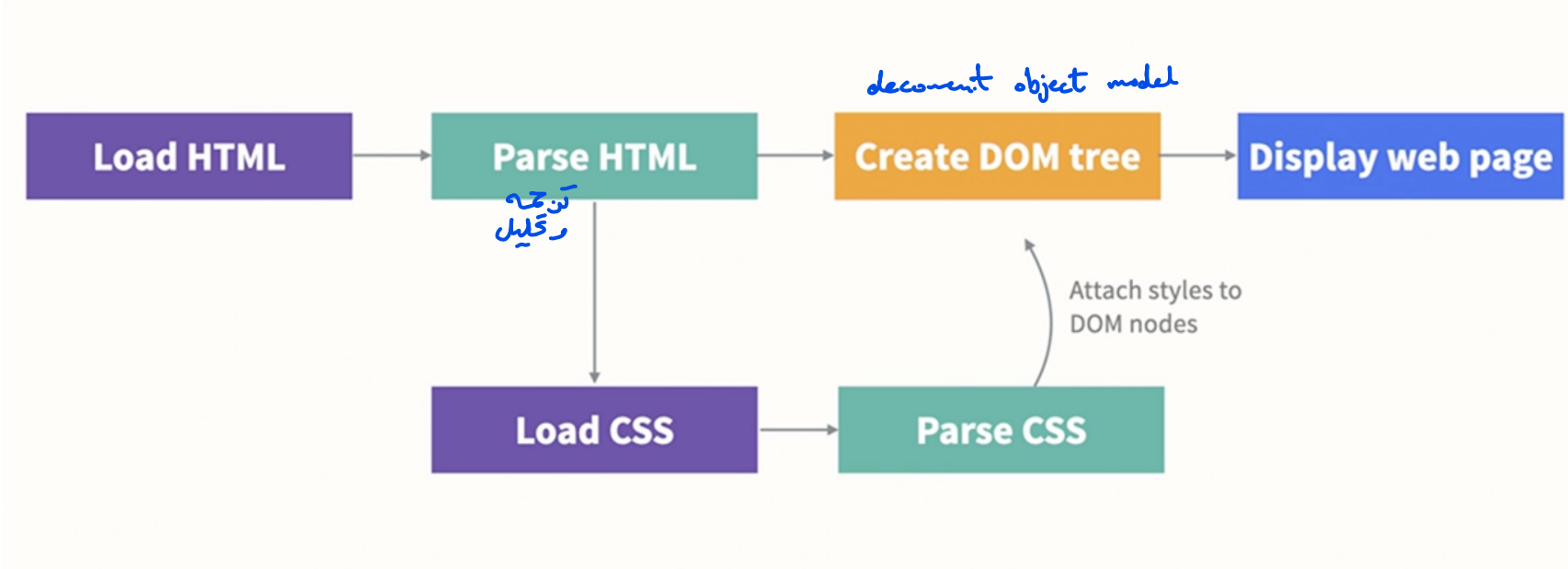
selector: Specifies the HTML element(s) to style

property: The name of the CSS style

value: The value for the CSS style

What is CSS?

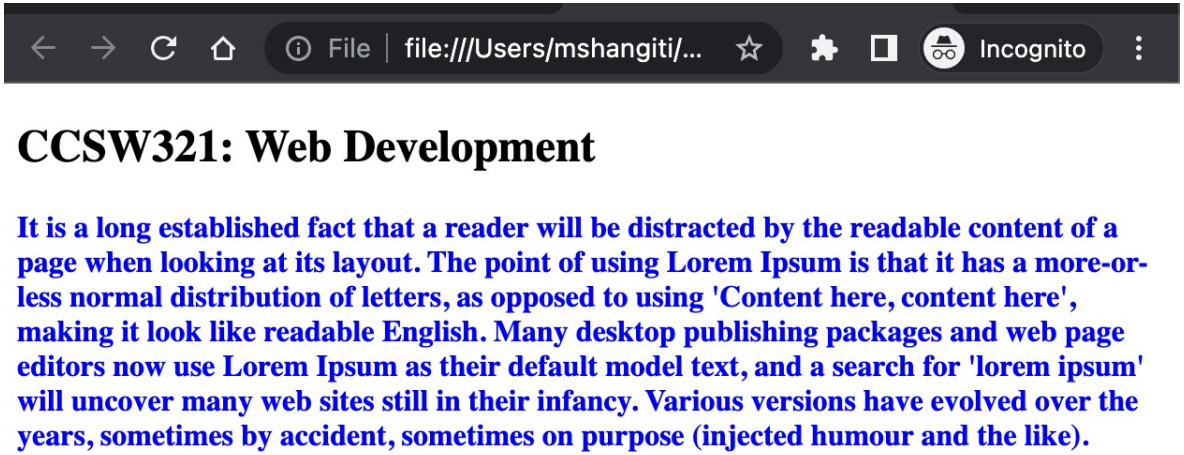
- How does a browser load a page?



CSS Example

```
p {
   color: blue;
   font-weight: bold;
}
```

"All **<p>** elements on the page should be blue and bold"



How to add CSS to a page

- Ways to include CSS

Inline Styles

```
<p style="color: red; margin-left: 20px"> This is a paragraph</p>
```

Embedded Styles

التغيرات في ال head

```
<head>
  <style type="text/css">
  body {background-color: red}
  p {margin-left: 20px}
  </style>
</head>
```

External Styles the best

```
<head>
  <link rel="stylesheet" type="text/css“ href="mystyle.css">
</head>
```

Advantages of external CSS

- **External CSS** is the recommended approach. **Why?**
  - Separate content from presentation.
  - Define appearance once and re-use for different pages.
  - Easier modification: When changes to the styles are required, you need to modify only a single CSS file to make style changes across all the pages that use those styles. This concept is sometimes known as skinning.

Software Engineering Observation 4.1

Inline styles do not truly separate presentation from content. To apply similar styles to multiple elements, use embedded style sheets or external style sheets, introduced later in this chapter.

CSS Validation

- CSS validation is the process of checking whether a CSS file follows the rules and syntax of the CSS language.
- CSS validation helps ensure that your CSS code is free of errors and that it will work correctly across different browsers and devices.
- CSS validation can help catch common errors, such as misspelled property names or invalid values, before they cause problems on your website or web application.

CSS Validation

- CSS validation can also help improve the performance of your website or web application, as it can identify and remove unnecessary or redundant CSS code.
- One popular CSS validator is the W3C CSS Validation Service, which checks CSS files against the official CSS specification published by the World Wide Web Consortium (W3C).
- https://jigsaw.w3.org/css-validator/

CSS Example

```
1  <!DOCTYPE html>
2
3  <!-- Fig. 4.3: embedded.html -->
4  <!-- Embedded style sheet. -->
5  <html>
6     <head>
7        <meta charset = "utf-8">
8        <title>Embedded Style Sheet</title>
9
10       <!-- this begins the style sheet section -->
11       <style type = "text/css">
12          em       { font-weight: bold;
13                     color: black; }
14          h1       { font-family: tahoma, helvetica, sans-serif; }
15          p        { font-size: 12pt;
16                     font-family: arial, sans-serif; }
17          .special { color: purple; }
18       </style>
19    </head>
```

class selector

Fig. 4.3 | Embedded style sheet. (Part 1 of 3.)

CSS Example

```
20      <body>
21         <!-- this attribute applies the .special style class -->
22         <h1 class = "special">Deitel & Associates, Inc.</h1>
23
24         <p>Deitel & Associates, Inc. is an authoring and
25            corporate training organization specializing in
26            programming languages, Internet and web technology,
27            iPhone and Android app development, and object
28            technology education.</p>
29
30         <h1>Clients</h1>
31         <p class = "special"> The company's clients include many
32            <em>Fortune 1000 companies</em>, government agencies,
33            branches of the military and business organizations.</p>
34      </body>
35   </html>
```

Fig. 4.3 | Embedded style sheet. (Part 2 of 3.)

CSS Example

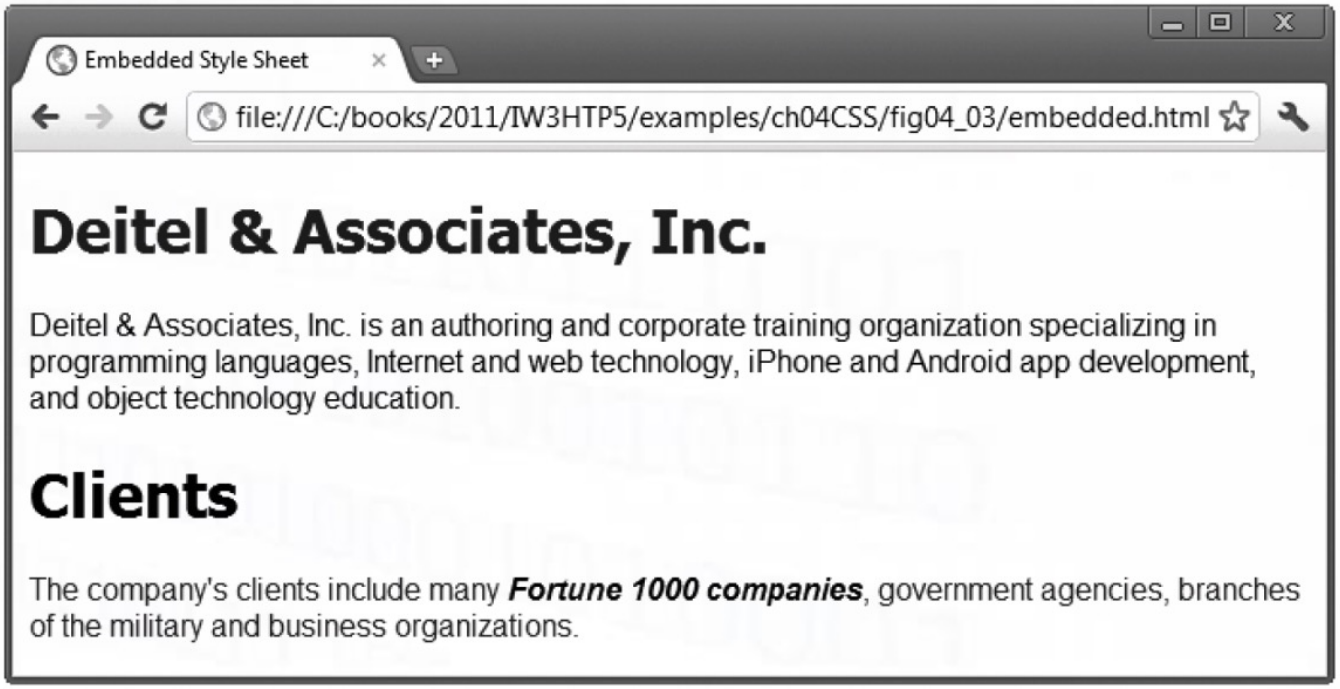


Fig. 4.3 | Embedded style sheet. (Part 3 of 3.)

CSS Selectors and Properties

- Universal Selector: Matches to all elements, of any type.

```
/* Applies a border to all elements*/
* {
  border: 1px solid black;
}
```

- Element/Type Selector: Matches the specified element.

```
/* Changes the text color of all <h1>
elements to the color red */
h1 {
  color: red;
}
```

CSS Selectors and Properties

- Class Selector: Matches to all elements with the specified class.

.special

```
<p class="fancy intro">Fancy intro paragraph.</p>
<p class="fancy">Fancy paragraph.</p>
<p>Regular paragraph.</p>

.fancy { ... } /* only fancy styles */
.intro { ... } /* only intro styles */

/* only applies if "fancy" AND "intro" are present */
.fancy.intro { ... }
```

Note that we can apply multiple classes to the same element

CSS Selectors and Properties

- ID Selector: Matches to the element with the specified ID. في الصفحة مرة وحده بس
  - ID value must be unique per page.

```
<div id="container">
  <p>Paragraph in a header.</p>
</div>

#container {
text-align: center;
}
```

الكلمة فالنص ←

Note on ID uniqueness: It is incorrect to use the value ‘container’ again within the same page

CSS Selectors and Properties

```
<h1 id="title">Homework</h1>
<em class="hw">HW0</em> is due Friday.<br/>
<em class="hw">HW1</em> goes out Monday.<br/>
<em>All homework due at 11:59pm.</em>
```

```
#title {
  color: purple;
}

.hw {
  color: hotpink;
}
```

Homework

HW0 is due Friday.
HW1 goes out Monday.
All homework due at 11:59pm.

CSS Selectors and Properties

- **CSS Combinators**: Explain the relationship between selectors.

Syntax	combinator	Example	Explanation
space	Descendant Selector	div p{color:blue}	This CSS targets ALL p elements that are descendants of a div element and sets their text color to blue.
>	Child Selector	div > p{color:red}	This CSS targets ALL the p element that are a direct child of the div element and sets their text color to red.
+	Adjacent Selector	div + p{color:green}	This CSS targets the FIRST p element that comes immediately after the div element and sets its color to green.
~	General Sibling Selector	div ~ p{color:yellow}	This CSS targets ALL p elements that come after the div element and sets their color to blue.

CSS Selectors and Properties

- Combinators Example: Descendant Selector

CSS

```
div span {
  color: blue;
}
```

HTML

```
<div>
     <span>I Match! </span>
     <ul>
          <li><span> I match, too! </span></li>
     </ul>
</div>
     <span> I do NOT match </span>
```

div span{} = ALL span elements that are descendants of a div element.

The browser would find any span elements nested inside of a div element, regardless of how many levels deep down that span tag is.

CSS Selectors and Properties

- Combinators Example: Child Selector

CSS

```
div > span {
  color: blue;
}
```

HTML

```
<div>
     <span>I Match! </span>
     <ul>
          <li><span> I do NOT match </span></li>
     </ul>
     <span>I Match too! </span>
</div>
     <span> I do NOT match </span>
```

div > span{} = ALL the span elements that are a **direct** child of the div element.

CSS Selectors and Properties

- Combinators Example: Adjacent Selector

CSS

```
div + span {
  color: blue;
}
```

HTML

```
<div>
     <span> I do NOT match </span>
     <ul>
          <li><span> I do NOT match </span></li>
     </ul>
     <span> I do NOT match </span>
</div>
  ↳ <span> I Match! </span>
     <span> I do NOT match </span>
```

div + span{} = The FIRST span element that comes immediately after the div element

CSS Selectors and Properties

- Combinators Example: General Sibling Selector

CSS

```
div ~ span {
  color: blue;
}
```

HTML

```
<div>
     <span> I do NOT match </span>
     <ul>
          <li><span> I do NOT match </span></li>
     </ul>
     <span> I do NOT match </span>
</div>
     <span> I Match! </span>
     <span> I Match too! </span>
```

div ~ span{} = This CSS targets ALL span elements that come after the div

CSS Selectors and Properties

- There are over 500 CSS properties! Here are a few Commonly used properties:

```
h1{
    /* font-* properties */
     font-size: 40px;
     font-family: 'Georgia';
     font-weight: bold;
     font-style: italic;

     /* text-* properties */
     text-decoration: underline;
     text-align: center;

     /* other properties */
     color:green;
     background-color: lightblue;
     border:2px dashed red;
}
```

HTML

```
<h1>Hello CCSW321!</h1>
```

Hello CCSW321!

CSS Selectors and Properties

- Font-size property specifies the size used to render the font.
  - You can specify a point size or a relative value such as xx-small, x-small, small, smaller, medium, large, larger, x-large and xx-large.
  - Relative font-size values are preferred over points, because an author does not know the specific measurements of each client's display.
  - Relative values permit more flexible viewing of web pages.
    - For example, users can change font sizes the browser displays for readability.

Relative and absolute measurements

- **Relative length** measurements: افضل من اي تحت
  - px (pixels – size varies depending on screen resolution)
  - em (usually the height of a font's uppercase M)
  - ex (usually the height of a font's lowercase x)
  - Percentages (of the font's default size)

- **Absolute-length** measurements (units that do not vary in size):
  - in (inches)
  - cm (centimeters)
  - mm (millimeters)
  - pt (points; 1 pt = 1/72 in)
  - pc (picas; 1 pc = 12 pt)

Good Programming Practice 4.1

Whenever possible, use relative-length measurements. If you use absolute-length measurements, your document may not scale well on some client browsers (e.g., smartphones).

CSS Selectors and Properties

- Font-family property specifies the name of the font to use.
  - Generic font families allow authors to specify a type of font instead of a specific font, in case a browser does not support a specific font.

Generic font families	Examples
serif	times new roman, georgia
sans-serif	arial, verdana, futura
cursive	script
fantasy	critter
monospace	courier, fixedsys

Fig. 4.5 | Generic font families.

CSS Selectors and Properties

- Font-weight property specifies the "boldness" of text.
- Possible values are:
  - Bold
  - normal (the default)
  - bolder (bolder than bold text)
  - lighter (lighter than normal text)
  - Boldness also can be specified with multiples of 100, from 100 to 900 (e.g., 100, 200, …, 900). Text specified as normal is equivalent to 400, and bold text is equivalent to 700

CSS Selectors and Properties

- Font-style Property: Allows you to set text style. Possible values to none, italic or oblique.
- Text-decoration property applies decorations to text in an element. Possible values are: underline, overline, line-through, blink.
- Text-align property sets text alignment. Possible values are left, right, center.

CSS Selectors and Properties

- Color property sets text color
- Color names and hexadecimal codes may be used as the color property value.

Color name	Value	Color name	Value
aqua	#00FFFF	navy	#000080
black	#000000	olive	#808000
blue	#0000FF	purple	#800080
fuchsia	#FF00FF	red	#FF0000
gray	#808080	silver	#C0C0C0
green	#008000	teal	#008080
lime	#00FF00	yellow	#FFFF00
maroon	#800000	white	#FFFFFF

Fig. 4.2 | HTML standard colors and hexadecimal RGB values.

CSS Selectors and Properties

- Background-color property sets the background color of an element. It can be set to any valid CSS color value, including color names, RGB values, HEX codes, HSL values, and more.
- Border property sets the border of an element. It can be used to set the width, style, and color of a border. It is an example of a shorthand property which can set multiple properties using one.

CSS Selectors and Properties

- Longhand and Shorthand properties
  - **Shorthand property** is a property that takes the values for other sets of CSS properties, e.g., border, padding, margin, etc.
    - The **border** property is a shorthand property since we can assign the value for border-width, border-style and border-color using the border property alone.
  - **Longhand property** is the individual property that can be used in place of the shorthand property. For example, the border-style property or the border-width property.

CSS Selectors and Properties

Basically,

```
1 | border: 1px solid blue;
```

Shorthand property

is the same as:

```
1 | border-width: 1px;
2 | border-style: solid;
3 | border-color: blue;
```

longhand properties

CSS Selectors and Properties

- Selectors Summary:

Example	Description
p	All **<p>** elements
.abc	All elements with the **abc class**, i.e. **class="abc"**
#abc	Element with the **abc id**, i.e. **id="abc"**
p.abc	**<p>** elements with **abc class**
p#abc	**<p>** element with **abc id** (**p** is redundant)
div strong	**** elements that are descendants of a ** **
h2, div	**<h2>** elements and ** ** يتطبق على الاثنين

CSS Selectors and Properties

- How to describe a selector?

```
#main li.important strong {
    color: red;
}
```

- **Read** from **right to left**:
  - tags that are children of <li> tags that have an "important" class that are children of the element with the "main" id

CSS Selectors and Properties

- Two common issues (misunderstanding of selectors)

```
p.abc     vs    p .abc

p .abc    vs    p, .abc
```

They look similar, but they are different!

- **Issue 1**: A <p> element with the abc class vs an element with the abc class that descends from <p>
- **Issue 2:** An element with the abc class that descends from <p> vs all <p> elements and all elements with the abc class

CSS Selectors and Properties

- pseudo-classes: special keywords you can append to selectors, specifying a state or property of the selector

Syntax	Explanation
a	All anchor tags (links) in all states
a:visited	A visited link
a:link	An unvisited link
a:hover	The style when you hover over a link
a:active	The style when you have "activated" a link (downclick)

Common Programming Error 4.1

Including a space before or after the colon separating a pseudo-class from the name of the element to which it's applied prevents the pseudo-class from being applied properly.

CSS Example

```
1  <!DOCTYPE html>
2
3  <!-- Fig. 4.6: advanced.html -->
4  <!-- Inheritance in style sheets. -->
5  <html>
6     <head>
7        <meta charset = "utf-8">
8        <title>More Styles</title>
9        <style type = "text/css">
10          body     { font-family: arial, helvetica, sans-serif; }
11          a.nodec  { text-decoration: none; }
12          a:hover  { text-decoration: underline; }
13          li em    { font-weight: bold; }
14          h1, em   { text-decoration: underline; }
15          ul       { margin-left: 20px; }
16          ul ul    { font-size: .8em; }
17       </style>
18    </head>
```

Fig. 4.6 | Inheritance in style sheets. (Part 1 of 4.)

CSS Example

```
19      <body>
20         <h1>Shopping list for Monday:</h1>
21
22         <ul>
23            <li>Milk</li>
24            <li>Bread
25               <ul>
26                  <li>white bread</li>
27                  <li>Rye bread</li>
28                  <li>Whole wheat bread</li>
29               </ul>
30            </li>
31            <li>Carrots</li>
32            <li>Yogurt</li>
33            <li>Pizza <em>with mushrooms</em></li>
34         </ul>
35
36         <p><em>Go to the</em>
37            <a class = "nodec" href = "http://www.deitel.com">
38               Grocery store</a>
39         </p>
40      </body>
41   </html>
```

Fig. 4.6 | Inheritance in style sheets. (Part 2 of 4.)

CSS Example

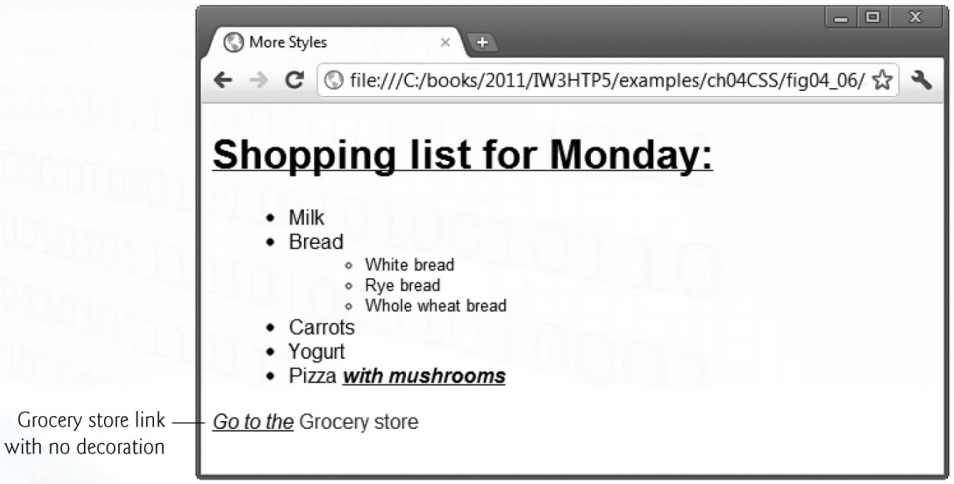


Fig. 4.6 | Inheritance in style sheets. (Part 3 of 4.)

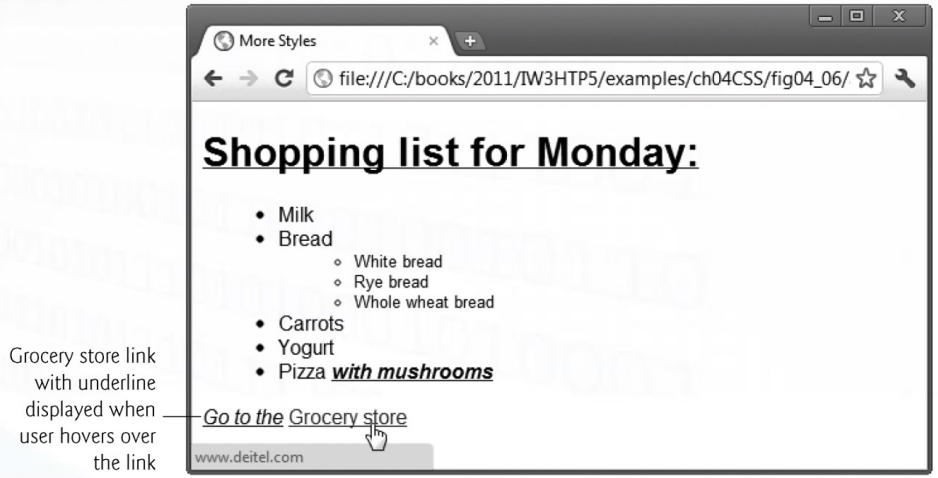


Fig. 4.6 | Inheritance in style sheets. (Part 4 of 4.)

CSS Inheritance

- CSS inheritance is the process by which properties of a parent element are passed down to its child elements.
- When a child element does not have a value set for a particular property, it will inherit the value of that property from its parent element.
- It's important to note that not all CSS properties are inherited by default. For example, box model properties (e.g., width, height, padding, margin) do not inherit by default, and must be set explicitly on each element.
- There's no golden rule for what properties are inherited or not; the inheritance behavior is defined in the CSS spec for each property.
  - Full CSS specs: https://www.w3.org/Style/CSS/

CSS Inheritance

- Generally, text-related properties are inherited and layout-related properties are not. Text-related properties include font properties (e.g., font-family, font-size, font-weight) and text properties (e.g., color, text-align, line-height).
- Understanding how CSS inheritance works can help you write more efficient and maintainable CSS code, as it allows you to apply styles to multiple elements at once by targeting a common parent element.

CSS Inheritance

Instead of selecting all elements individually:

```
a, h1, p, strong {
  font-family: Helvetica;
}
```

You can style the parent and the children will inherit the styles.

```
body {
  font-family: Helvetica;
}
```

You can override style for specific child elements via specificity:

```
h1,h2 {
  font-family: Consolas;
}
```

CSS Example

```
1  <!DOCTYPE html>
2
3  <!-- Fig. 4.17: dropdown.html -->
4  <!-- CSS drop-down menu. -->
5  <html>
6     <head>
7        <meta charset = "utf-8">
8        <title>
9           Drop-Down Menu
10       </title>
11       <style type = "text/css">
12          body             { font-family: arial, sans-serif }
13          nav              { font-weight: bold;
14                             color: white;
15                             border: 2px solid royalblue;
16                             text-align: center;
17                             width: 10em;
18                             background-color: royalblue; }
19          nav ul           { display: none;
20                             list-style: none;
21                             margin: 0;
22                             padding: 0; }
23          nav:hover ul     { display: block }
```

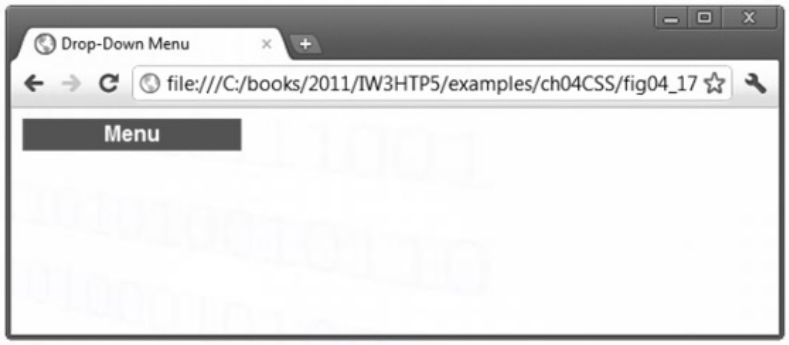
Fig. 4.17 | CSS drop-down menu. (Part 1 of 5.)

```
24          nav ul li        { border-top: 2px solid royalblue;
25                             background-color: white;
26                             width: 10em;
27                             color: black; }
28          nav ul li:hover  { background-color: powderblue; }
29          a                { text-decoration: none; }
30       </style>
31    </head>
32    <body>
33       <nav>Menu
34          <ul>
35             <li><a href = "#">Home</a></li>
36             <li><a href = "#">News</a></li>
37             <li><a href = "#">Articles</a></li>
38             <li><a href = "#">Blog</a></li>
39             <li><a href = "#">Contact</a></li>
40          </ul>
41       </nav>
42    </body>
43 </html>
```

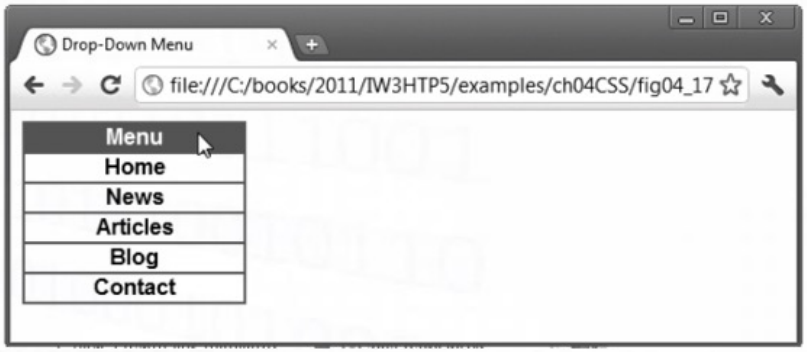
Fig. 4.17 | CSS drop-down menu. (Part 2 of 5.)

CSS Example

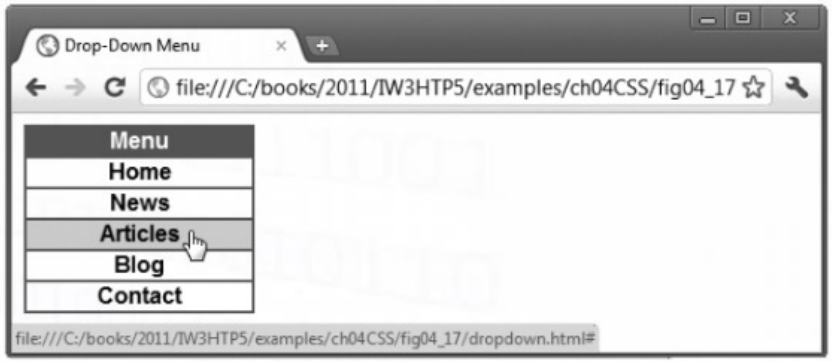
a) A collapsed menu



b) A drop-down menu is displayed when the mouse cursor is hovered over **Menu**



c) Hovering the mouse cursor over a menu link highlights the link



Conflicting Styles

- When multiple CSS rules target the same element, conflicts can occur. CSS has a set of rules for resolving conflicts:
  - Inline styles have the highest priority.
  - Embedded Styles defined in the head section of the HTML document come next.
  - External Styles defined in an external CSS file come next.
  - Browser default styles come last.

Conflicting Styles

- When styles collide, the most specific rule wins (specificity).
  - ID and class selectors are considered more specific than type selectors.
  - ID selectors are considered more specific than Class selectors.

```
div strong { color: red; }
    strong { color: blue; }

<div>
    <strong>What color am
I?</strong>
</div>
```

Answer: It will be RED

الاول

```
#success   { color: green;}
.info      { color: blue;}
div strong {color: red;}
<div>
    <strong class='info'
    id='success'>What color am
     I?</strong>
</div>
```

Answer: It will be GREEN

الأولويه للـ ID

Conflicting Styles Example

- What color will the text in the paragraph be?

```
<!DOCTYPE html>
<html>
<head>
<style>
  p {
     color: green;
  }
</style>
</head>
<body>
    <p style='color:red'>This is a paragraph.</p>
</body>
</html>
```

Inline ← الأولوية له

Answer: It will be RED

Why? The inline CSS will overwrite the embedded CSS

Tips for development

- **Tip1**: Learn when to use Class Selector vs. ID Selector:
  - The class selector can be assigned to multiple elements per page (reusable), whereas the ID selector is assigned to a single element per page (unique).
  - ID selectors are used for elements that occur once in a page, usually to help build boxes of content in the page, e.g., #main-content, #side.
  - The ID attribute is also more frequently used with JavaScript than CSS.
  - Class selectors are the more commonly used selectors. They are mostly used to define general reusable rules, e.g., changing the color of the text based on the message type (.success, .error, .warning, etc).

Tips for development

- **Tip2**: Use meaningful class and ID names

```
<html>

<head>...</head>

<body>
    <div id="box1">
        <section id="section1">
            <h2 class="main-heading">A Heading</h2>
            <p class="body-copy">Some bit of content.<
        </section>
        <div id="section2">
            <h2 class="main-heading">Sidebar</h2>
            <p class="body-copy">More content.</p>
            <p class="emphasis">This is important text
            <h3 class="red-text">Subheader</h3>
            <p class="big-text">Author's Name</p>
            <p class="body-copy">Some details</p>
        </div>
    </div>
</body>

</html>
```

```
<html>

<head>...</head>

<body>
    <div id="wrapper">
        <section id="main-content">
            <h2>A Heading</h2>
            <p>Some bit of content.</p>
        </section>
        <aside id="sidebar">
            <h2>Sidebar</h2>
            <p>More content.</p>
            <p><strong>This is important text!</strong></p>
            <h3>Subheader</h3>
            <p class="author-name">Author's Name</p>
            <p class="author-info">Some details</p>
        </aside>
    </div>
</body>

</html>
```

- Section1 to main-content.
- body-copy to author-info.

- Reduced class use to minimum.
  - (e.g., removed main-heading, body-copy).

Tips for development

- **Tip3**: Use consistent HTML structure
  - Inconsistent HTML = Many different rules for the same task

  - Consistent HTML = one rule, multiple usage = better CSS.

Tips for development

- **Tip4**: Structure and design go hand in hand!

Discouraged:

```
<h1 class="hw">Homework 0</h1>
<p class="hw">Due Fri</p>
<p class="hw">Late cutoff Sun</p>
```

vs

Preferred:

```
<div class="hw">
  <h1>Homework 0</h1>
  <p>Due Fri</p>
  <p>Late cutoff Sun</p>
</div>
```

Instead of applying a class to several adjacent elements, wrap the group in a ** ** container and style the contents via descendent selectors.

Any questions?
Please feel free to raise your hands and ask.