



# CS472 Web Programming

## Lecture 2: CSS



---

Except where otherwise noted, the contents of this document are Copyright 2012 Marty Stepp, Jessica Miller, Victoria Kirst and Roy McElmurry IV. All rights reserved. Any redistribution, reproduction, transmission, or storage of part or all of the contents in any form is prohibited without the author's expressed written permission. Slides have been modified for Maharishi University of Management Computer Science course CS472 in accordance with instructors agreement with authors.

## Maharishi University of Management - Fairfield, Iowa © 2016



All rights reserved. No part of this slide presentation may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying or recording, or by any information storage and retrieval system, without permission in writing from Maharishi University of Management.

# Wholeness Statement

---

In this lecture we introduce the basics of CSS as a way to give different visual styles to HTML elements, changing their preset default appearance. *The impulses of pure intelligence, the Veda, give rise to all aspects of the universe.*



# The bad way to produce styles

---

- ▶ Tags such as **strong**, **em**, **u**, and **font** are discouraged in strict HTML

```
<p>  
  <font face="Arial">Welcome to Greasy Joe's.</font>  
  You will <strong>never</strong>, <em>ever</em>,  
  <u>EVER</u> beat  
  <font size="+4" color="red">OUR</font> prices!  
</p>
```

Welcome to Greasy Joe's. You will **never**, *ever*, EVER beat **OUR** prices!

# Content vs. Presentation

---

- ▶ HTML is for content, the information on the page
- ▶ CSS is for presentation, how to display the page
- ▶ Keeping content separate from presentation is a very important web design principle
- ▶ If the HTML contains no styles, its entire appearance can be changed by swapping **.css** files
- ▶ <http://csszengarden.com/>



# Bad Practices, why?

---

## Embedding style sheets



```
<head>
  <style type="text/css">
    p { font-family: sans-serif; color:
red; }
    h2 { background-color: yellow; }
  </style>
</head>
```

## Inline styles: the style attribute



```
<p style="font-family: sans-serif; color: red;">This
is a paragraph</p>
```

---

Note: It has higher precedence than embedded or linked styles



# Cascading Style Sheets (CSS): `<link>`

---

- ▶ CSS describes the appearance and layout of information on a web page
  - ▶ (as opposed to HTML, which describes the content of the page)
- ▶ Can be embedded in HTML or placed into separate **.css** file (preferred)

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



# Main Point

---

- ▶ Cascading Style Sheets (CSS) provide a way for web developers to specify the appearance of content on the page by using selectors to specify the element(s), and then the values that properties of those elements (like font or color) should have. It is a best practice to keep CSS in a separate file, doing so makes your pages more flexible.

# Basic CSS rule syntax

---

- ▶ A **CSS** file consists of one or more rules
- ▶ A rule's selector specifies HTML element(s) and applies style properties
- ▶ The **\*** selector, selects all elements
- ▶ To add a comment we use: **/\* \*/**

```
selector {  
    property: value;  
    property: value; ...  
}  
  
p {  
    font-family: sans-serif;  
    color: red;  
}
```



# CSS properties for colors

---

```
p {  
    color: white;  
    background-color: blue;  
}
```

This paragraph uses the style above.

property	description
color	color of the element's text
background-color	color that will appear behind the element

# Specifying colors

---

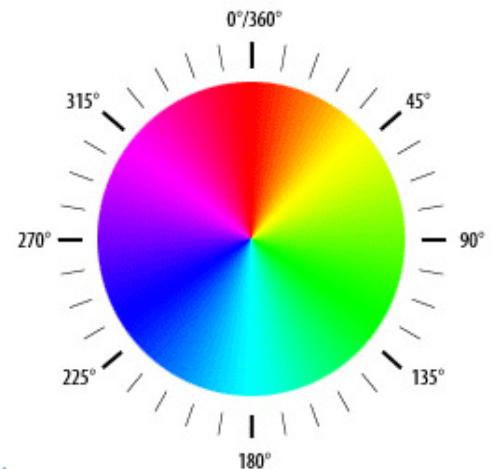
- ▶ **Color names:** aqua, black, blue, fuchsia, gray, green, lime, maroon, navy, olive, purple, red, silver, teal, white, yellow
- ▶ **RGB & RGBA codes:** red, green, and blue values from 0 to 255
- ▶ **HEX codes:** RGB values in base-16 from 00 (none) to FF (full)
- ▶ **HSL & HSLA codes:** HSL stands for hue, saturation, and lightness
  - ▶ Hue is degree on color wheel (from 0 to 360) - 0 (or 360) is red, 120 is green, 240 is blue.
  - ▶ Hsla demo <https://codepen.io/kman/pen/KwapPZ>
  - ▶ Google css color picker

```
h1 { color: red; }  
h2 { color: rgb(128, 0, 196); }  
h3 { color: rgba(128, 0, 196, 0.5); }
```

```
h4 { color: #FF8800; }
```

---

```
▶ 12 h5 { color: hsla(120, 60%, 70%, 0.3); }
```



# CSS properties for fonts

---

property	description	Values
font-family	which font will be used	serif or “Courier New”
font-size	how large the letters will be drawn	A unit value, percentage, or named value
font-style	used to enable/disable italic style	normal(default), italic, oblique
font-weight	used to enable/disable bold style	normal(default), bold, bolder,...
font	Sets all font properties	style weigh size family
<a href="#">Complete list of font properties</a>		

# CSS properties for fonts

---

```
h1{ /* which font will be used */  
    font-family: Georgia;  
}  
h2 { /* enclose multi-word font names in quotes */  
    font-family: "Courier New";  
}  
h3 { /* can specify multiple fonts from highest to  
lowest priority */  
    font-family: Garamond, "Times New Roman", serif;  
}
```

- ▶ If the first font is not found on the user's computer, the next is tried.
- ▶ Placing a generic font name at the end of your font-family value ensures that every computer will use a valid font
  - ▶ CSS generic font names: serif, sans-serif, cursive, fantasy, monospace
  - ▶ Serifed fonts easier to read on printed pages, hard to read on computer screens,

# font-size, font-weight, font-style

---



```
p {  
  /*    how large the letters will be drawn */  
  font-size:    14vw;  
  /*    used to enable/disable bold style */  
  font-weight:   bold;  
  /*    used to enable/disable italic style */  
  font-style:    italic;  
}
```

***CS472 WAP***  
***course has a lot***  
***of fun!***



# Size Units



---

- ▶ **Units:** pixels (**px**), point (**pt**), m-size (**em**)
- ▶ **pt** specifies number of points, where a point is 1/72 of an inch on screen
- ▶ **px** specifies a number of pixels on the screen
- ▶ **em** relative to the font-size of the element (2em means 2 times the size of the current font)
  
- ▶ **Vague font sizes:** xx-small, x-small, small, medium, large, x-large, xx-large, smaller, larger
  
- ▶ <https://webflow.com/blog/how-and-why-to-use-vh-and-vw-in-webflow>
- ▶ [https://www.w3schools.com/cssref/tryit.asp?filename=trycss\\_unit\\_vw](https://www.w3schools.com/cssref/tryit.asp?filename=trycss_unit_vw)



# CSS properties for text

---

Property	Description	Values
text-align	alignment of text within its element	left, center, right, justify
text-decoration	decorations such as underlining	underline, overline, line-through, blink, none
text-indent	Indent first line	a size(px, pt, %, em)
line-height	vertical size of each line	a size(px, pt, %, em)
letter-spacing	Horizontal gap between letters	a size(px, pt, %, em)
word-spacing	Horizontal gap between words	a size(px, pt, %, em)
text-overflow 	How to handle too-long text	clip, ellipsis, ellipsis-word
text-shadow 	A “drop shadow” next to text	Two distances(px, pt, %, em) plus an optional shaow color


# CSS properties for text



```
h2 {  
    /* Can also be overline, line-through, blink, or none. Effects can be  
    combined */  
    text-decoration: underline overline;  
    /* Shadow is specified as an X-offset, a Y-offset, and an optional color  
    */  
    text-shadow: -2px 5px gray;  
}  
  
p {  
    /* Alignment of text within its element, can be left, right, center, or  
    justify */  
    text-align: center;  
    /* Space between the lines in two paragraphs */  
    line-height: 30px;  
    /* Space between words in <p> elements should be 30 pixels */  
    word-spacing: 30px;  
    /* Indent the first line of all <p> elements with 50 pixels */  
    text-indent: 50px;  
}
```

# CSS properties for background

---

Property	Description	Values
background-color	color to fill background	A color
background-image	image to place in background	url(image URL)
background-position	placement of bg image within element	Two tokens for x/y as top, bottom, left, right, center, or a size(pt, px, %, em)
background-repeat	whether/how bg image should be repeated	repeat(default), repeat-x, repeat-y or no-repeat
background-attachment	whether bg image scrolls with page	scroll(default), fixed
background-size 	scaling of bg image	a size(pt, px, %, em), cover, or contain
background	shorthand to set all background properties	

# Background



```
body {
```

```
    /* image to place in background */
```

```
    background-image: url("../images/draft.jpg");
```

```
    /* How bg image should be repeated */
```

```
    /* can be repeat (default), repeat-x, repeat-y, or  
    no-repeat */
```

```
    background-repeat: repeat-x;
```

```
    /* placement of bg image within element */ /*  
    value consists of two tokens, can be top, left,  
    right, bottom, center, a percentage, or a length  
    value in px, pt, etc */
```

```
    background-position: 370px 20px;
```

```
}
```

# The `list-style-type` property



- ▶ **`none`**: No marker
- ▶ **`disc`** (default), **`circle`**, **`square`**
- ▶ **`decimal`**: 1, 2, 3, etc.
- ▶ **`decimal-leading-zero`**: 01, 02, 03, etc.
- ▶ **`lower-roman`**: i, ii, iii, iv, v, etc.
- ▶ **`upper-roman`**: I, II, III, IV, V, etc.
- ▶ **`lower-alpha`**: a, b, c, d, e, etc.
- ▶ **`upper-alpha`**: A, B, C, D, E, etc.
- ▶ **`lower-greek`**: alpha, beta, gamma, etc.

```
ol {  list-style-type:  lower-roman; }
```

i. first item  
ii. second item  
iii. third item

# Main Point

---

We discussed the CSS Properties for color, font, text, background, and lists, which are the basic properties used on almost every page.

<http://www.creativebloq.com/web-design/10-design-concepts-web-developers-need-know-1135255>

# Body styles

---

- ▶ To apply a style to the entire body of your page, write a selector for the body element
- ▶ Saves you from manually applying a style to each element

```
body {  
  font-size: 16px;  
}
```

# Inheriting styles



- ▶ Styles get inherited from containing elements
- ▶ **Not all properties are inherited** (notice link's color below)
  - ▶ E.g., margin

```
body {  
    font-family: sans-serif;  
    background-color: pink;  
}  
p {  
    color: green;  
}  
a {  
    text-decoration: underline;  
}  
h2 {  
    font-weight: bold;  
    text-align: center;  
}
```

## CS472 Web Programming

This [course](#) provides a systematic introduction to programming interactive and dynamic web applications.





# Styles that Conflict

---

```
/* select multiple elements separated by commas */
```

```
p, h1, h2 {
```

```
color: green;
```

```
background-color: grey;
```

```
}
```

```
/* when two styles set conflicting values for the same  
property, the latter style takes precedence */
```

```
h2 {
```

```
background-color: blue;
```

```
}
```

```
<p> This paragraph will use background color grey! </p>
```

```
<h2> This heading will use background color blue! </h2>
```

This paragraph will use background color grey!

This heading will use background color blue!

# Chrome Styles Pane

---

- ▶ Invoke the DevTools inspector on element
- ▶ displays styles for selected DOM node
- ▶ gray background are read-only
- ▶ exclamation marks mean that property name and/or value is not understood by the browser,
  - ▶ is ignored
- ▶ A style declaration may contain several properties with the same name. Only the last one takes effect, canceling the preceding ones.
  - ▶ will be struck out, like overridden properties.

# Cascading style sheets

---

It's called Cascading Style Sheets because the properties of an element cascade together in this order:

1. Browser's default styles (reference)
  2. External style sheet files (in a `<link>` tag)
  3. Internal style sheets (in a `<style>` tag in the page header)
  4. Inline style (the style attribute of an HTML element)
- 
- ▶ Basically, cascading works from top to bottom inside the page (Depends on your order – later styles will always override top ones).
  - ▶ Inheritance is how elements in the HTML markup inherit properties from their parent elements
  - ▶ cascade is how CSS declarations are applied, and how conflicting rules do or don't override each other.





# Override Rules

---

```
<p class="RedColor BlueColor">
```


**Lorem Ipsum**

```
</p>
```

```
#YellowColor {  
  color: yellow;  
}
```

```
.BlueColor {  
  color: blue;  
}
```

```
.RedColor {  
  color: red;  
}
```



Lorem Ipsum

# Style Specificity



- ▶ When multiple styles apply to an element and have the same origin precedence.
- ▶ The most specific one applies. If they have the same specificity, then the later one will be used.

```
<aside>  
  <p><em id="recent" class="awesome">Which awesome color?</em></p>  
</aside>
```

```
aside { color: gray; }  
p { color: green; }  
em { color: yellow; }  
.awesome { color: blue; }  
em.awesome { color: red; }  
#recent { color: black; }  
em#recent.awesome { color: orange; }
```

*Which awesome color?*

# Specificity and conflicts

---

- ▶ Specificity- decide which one should win when two or more rules conflict.
- ▶ Rules: each rule's overall selector is given a score based upon approximately the following rules. The rule with the highest score wins if there's a conflict.
  - ▶ Any HTML element mentioned in the rule scores 1 point
  - ▶ Any class mentioned in the rule scores 10 points
  - ▶ Any ID mentioned in the rule scores 100 points
- ▶ Examples:
  - ▶ p.banner - 11
  - ▶ div.box > p - 12
  - ▶ body #logo .box p.banner - 122

# The HTML **class** and **id** attribute

---

- ▶ **id attribute** allows you to give a unique ID to any element on a page
  - ▶ Each ID must be unique; can only be used once in the page
- ▶ **class attribute** is used to group some elements and give a style to only that group
  - ▶ unlike an id, a class can be reused as much as you like on the page

# class vs id examples



```
<p id="mission">Our mission is to provide the most</p>
<p class="special">See our spectacular spatula specials</p>
<p class="special shout">Today only, satisfaction guaranteed</p>
```

```
#mission {
  font-style: italic;
  color: #000000;
}
.special { /* any element with class="special" */
  background-color: yellow;
  font-weight: bold;
}
p.shout { /* only p elements with class="shout" */
  color: red;
  font-family: cursive;
}
```

*Our mission is to provide the most*

**See our spectacular spatula specials**

**Today only, satisfaction guaranteed**



## class naming

---

- ▶ focus on the semantics and meaning of the content vs appearance
- ▶ Bad example: `redtext`, `bigfont`
  - ▶ if change style later, it doesn't make sense to be called `redtext`.
- ▶ Good example:
  - ▶ **`warningMsg`**
  - ▶ **`errorMsg`**

# CSS pseudo-classes

## pseudo-elements



- ▶ A **pseudo-class** is used to define a special state of an element
  - ▶ Style an element when a user mouse's over it
  - ▶ Style visited and unvisited links differently
  - ▶ Style an element when it gets focus
- ▶ A CSS **pseudo-element** is used to style specified parts of an element
  - ▶ Style the first letter, or line, of an element
    - ▶ ::first-line, ::first-letter
  - ▶ Insert content (pseudo element) before, or after, the content of an element
    - ▶ ::before, ::after





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

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

```
/* double colon notation - ::pseudo-element versus :pseudo-  
class */
```

# CSS pseudo-classes pseudo-elements

---

class		description
:active		an activated or selected element
:focus		an element that has the keyboard focus
:hover		an element that has the mouse over it
:link		a link that has not been visited
:visited		a link that has already been visited
:nth-child( <i>expr</i> )		targets specific children of a given element
:first-child, :last-child		
:not( <i>selector</i> )		all elements that do not match the given CSS selector
::first-line		the first line of text inside an element
::first-letter		the first letter of text inside an element

# Examples pseudo-classes

---



```
/* unvisited link */
```

```
a:link { color: #FF0000; }
```

```
/* visited link */
```

```
a:visited { color: #00FF00; }
```

```
/* mouse over link */
```

```
a:hover { color: #FF00FF; }
```

```
/* click on a link */
```

```
a:active { color: #0000FF; }
```

More info and examples: [Pseudo-classes](#) and [Pseudo-elements](#)

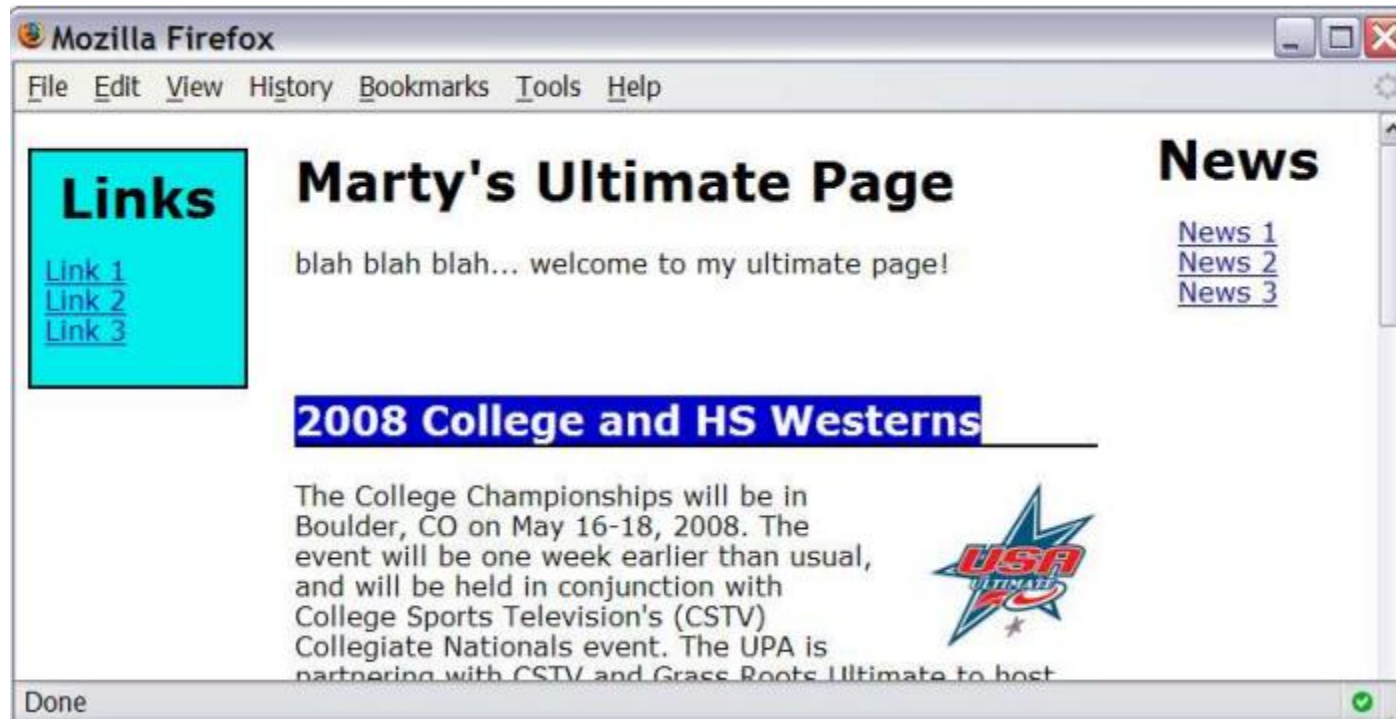
# add *content* to your website using CSS

---

- ▶ CSS has a property called content.
- ▶ only be used with the pseudo elements ::after and ::before.
- ▶ The ::after selector inserts something after the content of each selected element(s).
  - ▶ Use the content property to specify the content to insert.
  - ▶ Use the ::before selector to insert something before the content.
- ▶ [https://www.w3schools.com/cssref/tryit.asp?filename=trycss\\_sel\\_after](https://www.w3schools.com/cssref/tryit.asp?filename=trycss_sel_after)

# Motivation for page sections

- ▶ Want to be able to **style individual elements, groups of elements, sections of** <sub>text</sub> or of the page
- ▶ Want to create complex page layouts



# Sections of a page: `<div>` vs `<span>`

---

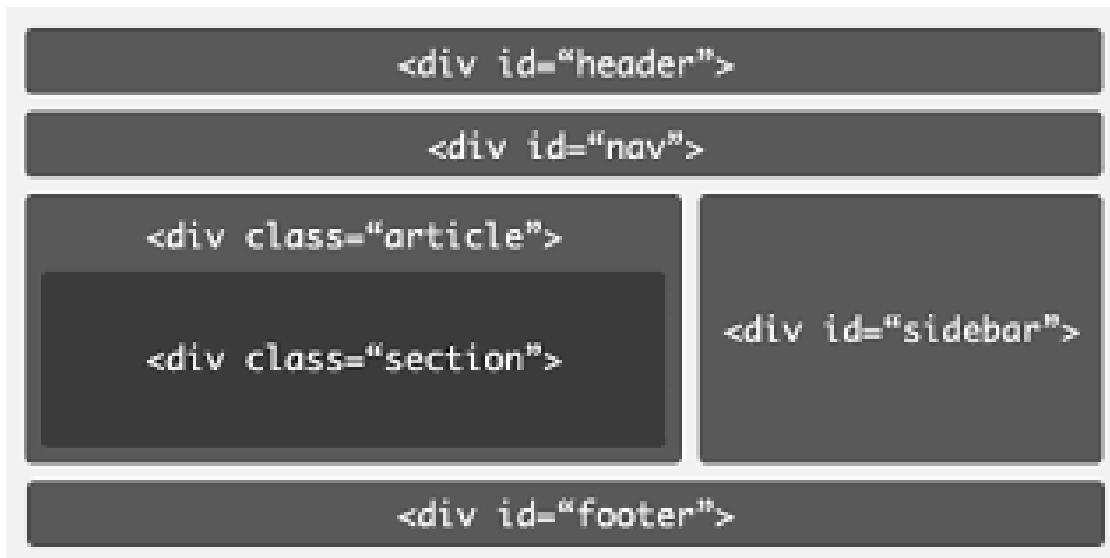
- ▶ `<div>` is a block element
- ▶ `<span>` is an inline element
- ▶ They have no onscreen appearance, but you can apply styles to them
- ▶ They carry no significant semantic meaning

```
<div class="shout">  
  <h2>Hello</h2>  
  <p class="special">See our specials!</p>  
  <p>We'll beat <span class="shout">all  
    prices!</span></p>  
</div>
```

# Review: HTML5 tags for page sections

---

- ▶ Serve the same purpose as `div` – more semantic and descriptive than `div`s
- ▶ Note in example, section can be section of an article or section of document containing articles





# CSS context selectors

---

`selector1 selector2 { properties }`

Applies the given properties to **selector2 only** if it is inside a selector1 on the page

`selector1 > selector2 { properties }`

- ▶ Applies the given properties to **selector2 only** if it is direct child of selector1 (in the DOM)
  - ▶ mnemonic: think of > as indicating a direct link (to a child)

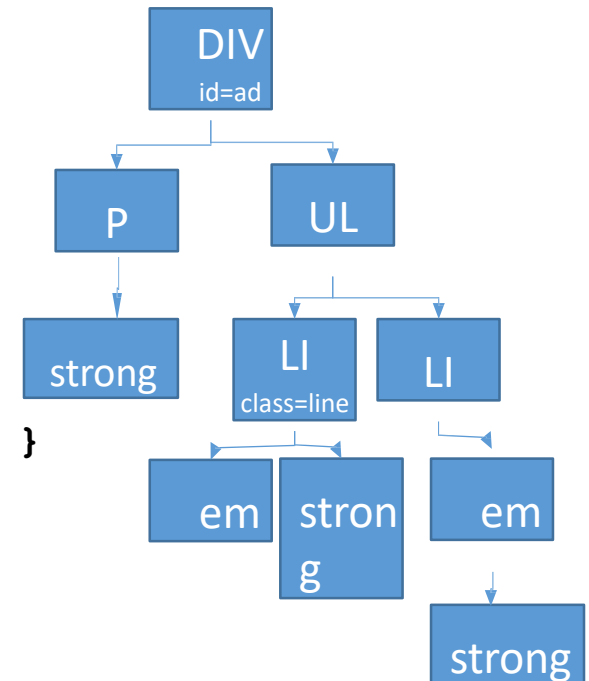
# Example



```
<div id="ad">
  <p>Shop at <strong>Hardwick's Hardware</strong></p>
  <ul>
    <li class="line"><em>The </em><strong>best</strong>
      prices!</li>
    <li><em><strong>Act while supplies last!</strong></em></li>
  </ul>
</div>
```

```
ul > li { background-color: blue; }
li strong { color: red; }
li > strong { color: green; }
#ad li.line strong { text-decoration: underline; }
```

See example: lesson3\_examples\contextselector.html,  
contextselectordirect.html



# Main Point

---

The `<div>` tag provides a generic block level element that can be used for any division or section of your page. The `<span>` tag provides a generic inline element for specifying any range of text inside a box. By using these tags, combined with CSS context selectors (direct child or ascendant) we can write powerful and reusable CSS rules.

# W3C CSS Validator

---

- ▶ Check your CSS to make sure it meets the official CSS specifications
- ▶ More picky than the web browser, which may render malformed CSS correctly

```
<p>  
  <a href="http://jigsaw.w3.org/css-validator/check/referer">  
      
  </a>  
</p>
```

# CONNECTING THE PARTS OF KNOWLEDGE WITH THE WHOLENESS OF KNOWLEDGE

## *Changing Appearances*

1. How a page is displayed is affected by both the HTML and the CSS
  2. Although every HTML tag has a default way of displaying, it can easily be changed with CSS and should never be the basis for using it. Instead use HTML tags based on meaning.
- 

3. **Transcendental consciousness** is the field that underlies all differences.
4. **Impulses within the Transcendental field:** the diversity of the universe arises as an expression of the unified field.
5. **Wholeness moving within itself:** In Unity Consciousness, one experiences that this unbounded diversity is the Self.

