Academic Year: 2020 – 2021

Department: **COMPUTER SCIENCE & ENGINEERING**

Course Code	Course Title	Core/ Elective	Prerequisite	Contact Hours			Total Hrs/ Sessions
				L	T	P	
18CS63	**WEB TECHNOLOGY AND ITS APPLICATIONS**	**Core**		**3**	**2**	**-**	**50**

Objectives

This course will enable students to

- Illustrate the Semantic Structure of HTML and CSS
- Compose forms and tables using HTML and CSS
- Design Client-Side programs using JavaScript and Server-Side programs using PHP
- Infer Object Oriented Programming capabilities of PHP
- Examine JavaScript frameworks such as jQuery and Backbone

Module – 1

Introduction to HTML, What is HTML and Where did it come from?, HTML Syntax, Semantic Markup, Structure of HTML Documents, Quick Tour of HTML Elements, HTML5 Semantic Structure Elements, Introduction to CSS, What is CSS, CSS Syntax, Location of Styles, Selectors, The Cascade: How Styles Interact, The Box Model, CSS Text Styling.

Module – 2

HTML Tables and Forms, Introducing Tables, Styling Tables, Introducing Forms, Form Control Elements, Table and Form Accessibility, Microformats, Advanced CSS: Layout, Normal Flow, Positioning Elements, Floating Elements, Constructing Multicolumn Layouts, Approaches to CSS Layout, Responsive Design, CSS Frameworks.

Module – 3

JavaScript: Client-Side Scripting, What is JavaScript and What can it do?, JavaScript Design Principles, Where does JavaScript Go?, Syntax, JavaScript Objects, The Document Object Model (DOM), JavaScript Events, Forms, Introduction to Server-Side Development with PHP, What is Server-Side Development, A Web Server's Responsibilities, Quick Tour of PHP, Program Control, Functions.

Module – 4

PHP Arrays and Superglobals, Arrays, $_GET and $_POST Superglobal Arrays,$_SERVER Array, $_Files Array, Reading/Writing Files, PHP Classes and Objects, Object-Oriented Overview, Classes and Objects in PHP, Object Oriented Design, Error Handling and Validation, What are Errors and Exceptions?, PHP Error Reporting, PHP Error and Exception Handling.

Module – 5

Managing State, The Problem of State in Web Applications, Passing Information via Query Strings, Passing Information via the URL Path, Cookies, Serialization, Session State, HTML5 Web Storage, Caching, Advanced JavaScript and jQuery, JavaScript Pseudo-Classes, jQuery Foundations, AJAX, Asynchronous File Transmission, Animation, Backbone MVC Frameworks, XML Processing and Web Services, XML Processing, JSON, Overview of Web Services.

List of Text Books

1. Randy Connolly, Ricardo Hoar, **"Fundamentals of Web Development",** 1st Edition, Pearson Education India. (**ISBN:**978-9332575271)

List of Reference Books

1) Robin Nixon, **"Learning PHP, MySQL &JavaScript with jQuery, CSS and**

2) Luke Welling, Laura Thomson, **"PHP and MySQL Web Development",** 5th Edition, Pearson Education, 2016. (**ISBN:**978-9332582736)

3) Nicholas C Zakas, **"Professional JavaScript for Web Developers",** 3rd Edition, Wrox/Wiley India, 2012. (**ISBN:**978-8126535088)

4) David Sawyer Mcfarland, **"JavaScript & jQuery: The Missing Manual",** 1st Edition, O'Reilly/Shroff Publishers & Distributors Pvt Ltd, 2014 (**ISBN:**978- 9351108078)

5) Zak Ruvalcaba Anne Boehm, **"Murach's HTML5 and CSS3"**, 3rdEdition, Murachs/Shroff Publishers & Distributors Pvt Ltd, 2016. (**ISBN:**978-9352133246)

List of URLs, Text Books, Notes, Multimedia Content, etc

1. https://www.w3schools.com/htmL/

2. https://html.com/

Course Outcomes	After studying this course, students will be able to:
CO1	Adapt HTML and CSS syntax and semantics to build web pages.
CO2	Construct , visually format tables and forms using HTML using CSS
CO3	Develop Client-Side Scripts using JavaScript and Server-Side Scripts using PHP to generate and display the contents dynamically.
CO4	Appraise the principles of object oriented development using PHP
CO5	Inspect JavaScript frameworks like jQuery and Backbone which facilitates developer to focus on core features.

Module – 1
Introduction to HTML

Brief History of HTML

- ARPANET of the late1960s
- Jump quickly to the first public specification of the HTML by Tim Berners-Lee in 1991
- HTML's codification by the World-Wide Web Consortium (better known as the **W3C**)in 1997.

HTML Syntax

- HTML is defined as a **markup language**.
- A markup language is simply a way of annotating a document in such a way to make the annotations distinct from the text being annotated.
- The term comes from the days of print, when editors would write instructions on manuscript pages that might be revision instructions to the author or copy editor. Markup.
- At its simplest, **markup**is a way to indicate information about the content
- This "information about content" in HTML is implemented via **tags** (elements).
- The markup in the previous slide consists of the red text and the various circles and arrows on the one page, and the little yellow sticky notes on the other.
- HTML does the same thing but uses textual tags.

What is the W3C?

- The W3C is the main standards organization for the World Wide Web.
- To promote compatibility the W3C produces **recommendations** (also called **specifications**).
- In 1998, the W3C turned its attention to a new specification called XHTML 1.0, which was a version of HTML that used stricter XML(Extensible Markup Language)syntax rules.

XHTML

The XML-based syntax rules for XHTML are pretty easy to Follows.

- lowercase tag names
- attributes always within quotes,
- And all elements must have a closing element (or be self- closing).

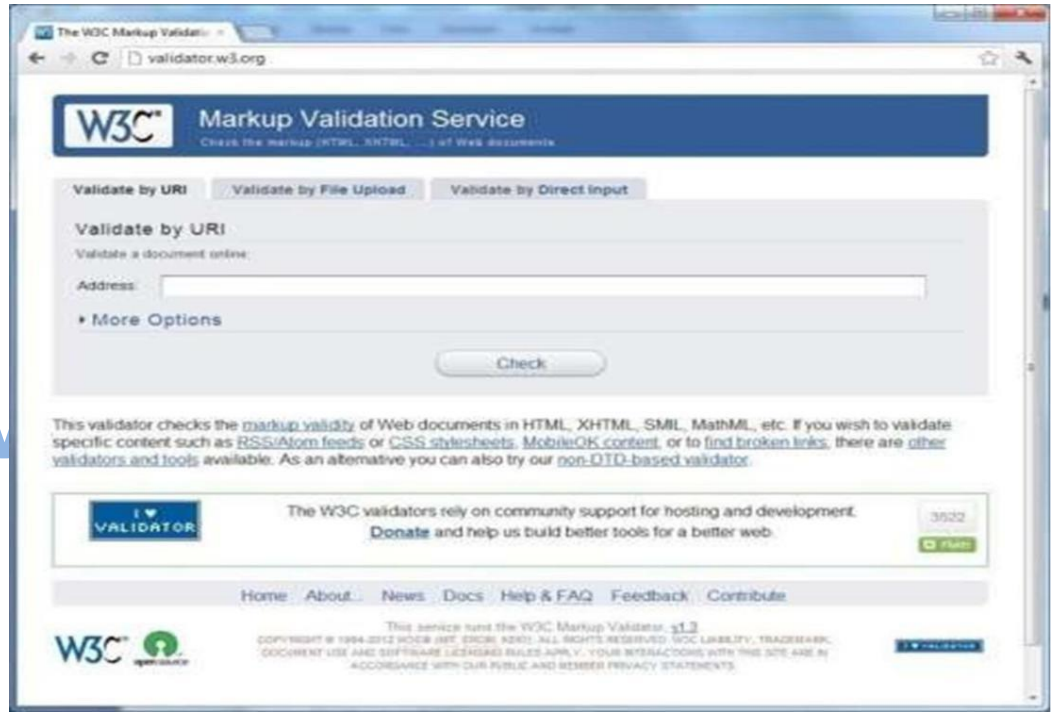
To help web authors, two versions of XHTML were created:

XHTML 1.0 Strict and **XHTML1.0 Transitional**.

- The **strict** version was meant to be rendered by a browser using the strict syntax rules and tag support described by the W3C XHTML1.0 Strict specification.
- The **transitional** recommendation is a more forgiving flavour of XHTML, and was meant to act as a temporary transition to the eventual global adoption of XHTML Strict.

Validators

A key part of the standards movement in the web development community of the 2000s was the use of **HTML Validators** as a means of verifying that a web page's markup followed the rules for XHTML transitional or strict.



- Backwards compatibility with HTML and XHTML1.0 was dropped.
- Browsers would become significantly less forgiving of invalid markup.
- At around the same time, a group of developers at Opera and Mozilla formed the **WHATWG** (Web Hypertext Application Technology Working Group) group within theW3C.
- This group was not convinced that the W3C's embrace of XML and its abandonment of backwards-compatibility was the best way forward for the web.

HTML5

By 2009, the W3Cstopped work on XHTML2.0 and instead adopted the work done by WHATWG and named it HTML5.

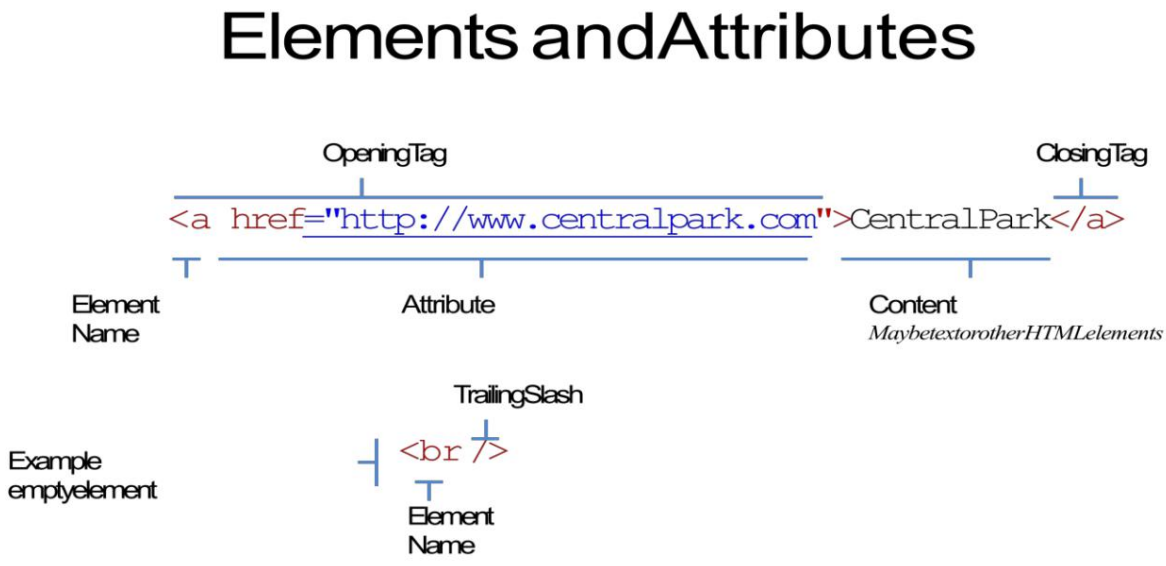
There are three main aims to HTML5:

- Specify unambiguously how browsers should deal with invalid markup.
- Provide an open, non-proprietary programming framework (via JavaScript) for creating rich web applications.
- Be backwards compatible with the existing web.
- While parts of the HTML5 are still being finalized, all of the major browser manufacturers have at least partially embraced HTML5.
- Certainly not all browsers and all versions support every feature of HTML5.

HTMLSYNTAX

Elements and Attributes

- **HTML documents** are composed of textual content and HTML elements.
- An **HTML element** can contain text, other elements, or be empty. It is identified in the HTML document by tags.
- HTML elements can also contain attributes. An **HTML attribute** is a name=value pair that provides more information about the HTML element.
- *In XHTML, attribute values had to be enclosed in quotes; in HTML5, the quotes are optional.*



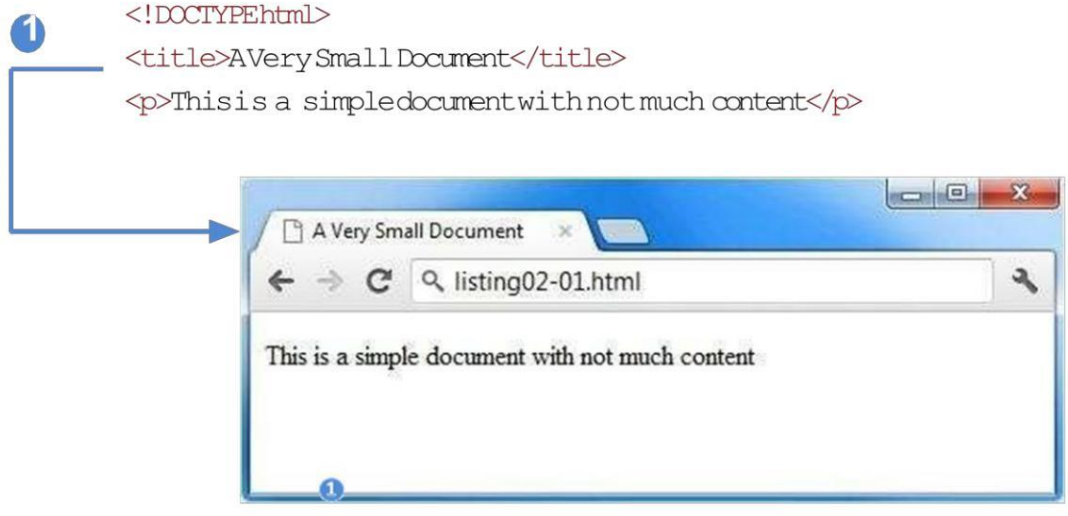
SEMANTICMARKUP

- Over the past decade, a strong and broad consensus has grown around the belief that HTML documents should **only** focus on the structure of the document.
- Information about how the content should look when it is displayed in the browser is best left to CSS (Cascading Style Sheets).
- As a consequence, beginning HTML authors are often counselled to create **semantic HTML** documents.
- That is, an HTML document should not describe how to visually present content, but only describe its content's structural semantics or meaning.

Eliminating presentation-oriented markup and writing semantic HTML markup has a variety of important advantages:

- **Maintainability**. Semantic markup is easier to update and change than web pages that contain a great deal of presentation markup.
- **Faster**. Semantic web pages are typically quicker to author and faster to download.
- **Accessibility**. Visiting a web page using voice reading software can be a very frustrating experience if the site does not use semantic markup.
- **Search engine optimization**. Semantic markup provides better instructions for search engines: it tells them what things are important content on the site.

STRUCTURE OF HTML



- The <title> element (Item) is used to provide a broad description of the content. The title is not displayed within the browser window. Instead, the title is typically displayed by the browser in its window and/or tab.

1) DOCTYPE

- Tells the browser (or any other client software that is reading this HTML document) what type of document it is about to process.
- Notice that it does not indicate what version of HTML is contained within the document: it only specifies that it contains HTML.

2) HTML,Head, and Body

- HTML5does not require the use of the <html>, <head>, and <body>.
- However, in XHTML they were required, and most web authors continue to use them.
- The<html> element is sometimes called the **root element** as it contains all the other HTML elements in the document.

QUICK TOUR OF HTML

1) Headings

- HTML provides six levels of heading (**h1, h2, h3**, …), with the higher heading number indicating a heading of less importance.
- Headings are an essential way for document authors use to show their readers the structure of the document.

2) Paragraphs

- Paragraphs are the most basic unit of text in an HTML document.
- Notice that the **<p>** tag is a container and can contain HTML and other **inline**
- **HTML elements**
- Inline HTML elements refers to HTML elements that do not cause a paragraph break but are part of the regular "flow" of the text.

3) Divisions

This ** ** tag is also a container element and is used to create a logical grouping of content

- The element has no intrinsic presentation.
- It is frequently used in contemporary CSS-based layouts to mark out sections.

4) Links

➢ Linksare created using the **<a>**element (the "a" standsfor anchor).

➢ Alink hastwo main parts: the destination and thelabel.

```
<a href="http://www.centralpark.com">Central Park</a>
```

Destination — Label(text)

```
<a href="index.html"><img src="logo.gif" /></a>
```

Label(image)

Types of Links

- You can use the anchor element to create a wide range of links:
- Links to external sites (or to individual resources such as images or movies on an external site).
- Links to other pages or resources within the current site.
- Links to other places with in the current page.
- Links to particular locations on another page.
- Links that are instructions to the browser to start the user's email program.
- Links that are instructions to the browser to execute a JavaScript function.

5) Images

✓ While the **<img>**tag is the oldest method for displaying an image, it is not the only way.

✓ For purely decorative images, such as background gradients and patterns, logos, border art, and so on, it makes semantic sense to keep such images out of the markup and in CSS where they more rightly belong.

✓ But when the images are content, such as in the images in a gallery or the image of a product in a product details page, then the <img> tag is the semantically appropriate approach.

6) Lists

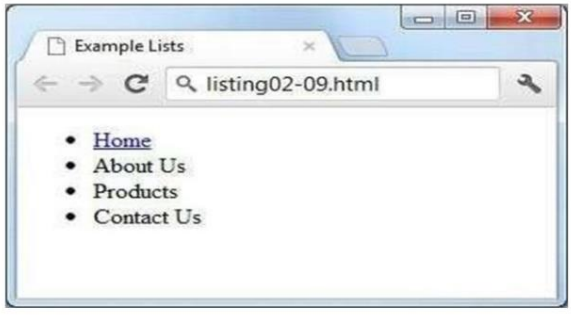
➢ **Unordered lists**. Collections of items in no particular order; these are by default rendered by the browser as a bulleted list.

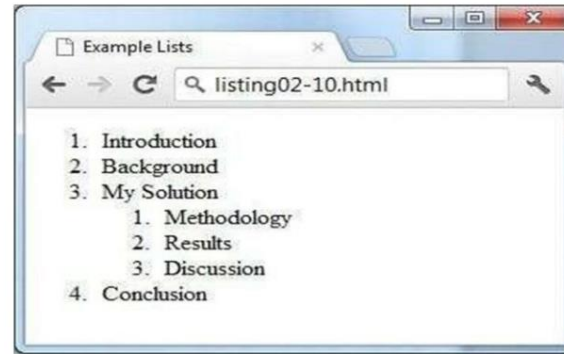
➢ **Ordered lists**. Collections of items that have a set order; these are by default rendered by the browser as a numbered list.

Notice that the list item element can contain other HTML elements

```
<ul>
   <li><a href="index.html">Home</a></li>
   <li>About Us</li>
   <li>Products</li>
   <li>Contact Us</li>
</ul>
```

```
<ol>
   <li>Introduction</li>
   <li>Background</li>
   <li>My Solution</li>
   <li>
      <ol>
         <li>Methodology</li>
         <li>Results</li>
         <li>Discussion</li>
      </ol>
   </li>
   <li>Conclusion</li>
</ol>
```





HTMLSEMANTIC ELEMENTS

1) Header and Footer

Most web site pages have are cognizable header and footer section.
Typical y the **header** contains

- the site logo
- title (and perhaps additional subtitles or tag lines)
- horizontal navigation links, and
- Perhaps one or two horizontal banners.

❑Both the HTML5 <header> and <footer> element can be used not only for *page* headers and footers, they can also be used for header and footer elements within other HTML5 containers, such as<article> or<section>.

```
<header>
   <img src="logo.gif" alt="logo" />
   <h1>Fundamentals of Web Development</h1>
   ...
</header>
<article>
   <header>
      <h2>HTML5 Semantic StructureElements
</h2>
      <p>By <em>Randy Connolly</em></p>
      <p><time>September 30, 2012</time></p>
   </header>
   ...
```

2) Figure and Figure Captions

- The W3C Recommendation indicates that the <figure> element can be used not just for images but for any type of *essential* content that could be moved to a different location in the page or document and the rest of the document would still make sense.
- The**<figure>**element should **not** be used to wrap every image.
- For instance, it makes no sense to wrap the site logo or non- essential images such as banner ads and graphical embellishments within <figure>elements.
- Instead, only use the <figure> element for circumstances where the image (or other content) has a caption and where the figure is essential to the content but its position on the page is relatively unimportant.

- The W3C Recommendation indicates that the <figure> element can be used not just for images but for any type of *essential* content that could be moved to a different location in the page or document and the rest of the document would still make sense.
- The**<figure>**element should **not** be used to wrap every image.
- For instance, it makes no sense to wrap the site logo or non- essential images such as banner ads and graphical embellishments within <figure>elements.
- Instead, only use the <figure> element for circumstances where the image (or other content) has a caption and where the figure is essential to the content but its position on the page is relatively unimportant.


```
<p>This photo was taken on October 22, 2011 with a Canon EOS 30D camera.</p>
<figure>
  <img src="images/central-park.jpg" alt="Central Park" /><br/>
  <figcaption>Conservatory Pond in Central Park</figcaption>
</figure>
<p>
It was a wonderfully beautiful autumn Sunday, with strong sunlight and
expressive clouds. I was very fortunate that my one day in New York was
blessed with such weather!
</p>
```

CSS1

CSS is a W3Cstandard for describing the **presentation (or appearance)** of HTML elements. With CSS, we can assign

- font properties,
- colors
- sizes
- borders
- background images
- Even the position of elements.

CSS is a language in that it has its own syntax rules.

CSS can be added directly to any HTML element (via the style attribute), within the **<head>** element, or, most commonly, in a separate ext file that contains only CSS.

Benefits of CSS

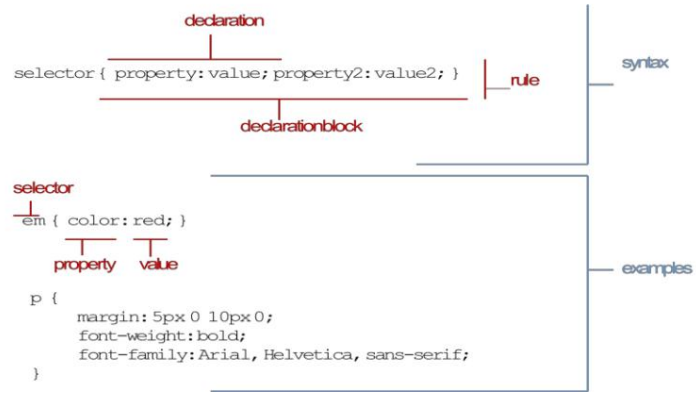
- The degree of formatting control in CSS is significantly better than that provided in HTML.
- Web sites become significantly more maintainable because all formatting can be centralized into one, or a small handful, of CSS files.
- CSS-driven sites are more accessible.
- A site built using a centralized set of CSS files for all presentation will also be quicker to download because each individual HTML file will contain fewer markups.
- CSS can be used to adopt a page for different output mediums.

CSS Versions

- W3Cpublished the CSS Level 1 Recommendation in 1996.
- A year later, the CSS Level 2 Recommendation (also more Succinctly labeled simply as CSS2)was published.
- Even though work began over a decade ago, an updated version of the Level 2 Recommendation, CSS2.1, did not become an official W3CRecommendation until June2011.
- And to complicate matters even more, all through the last decade (and to the present day as well), during the same time the CSS2.1 standard was being worked on, a different group at the W3Cwasworking on a CSS3draft.

CSS Syntax

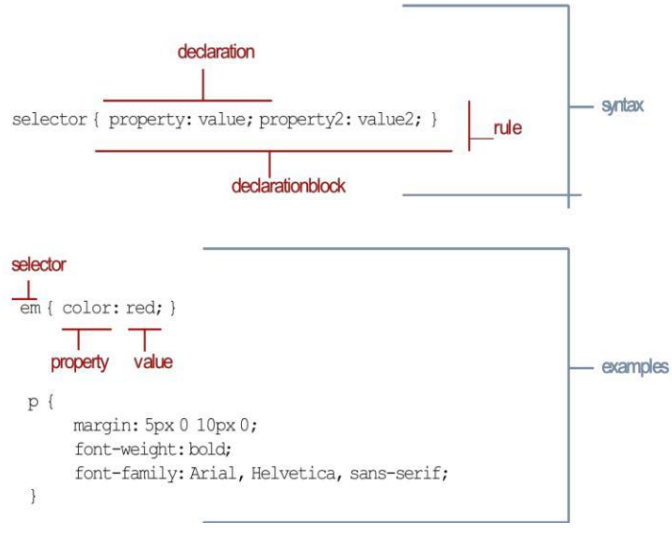
- A CSS document consists of one or more **stylerules**.
- A rule consists of a selector that identifies the HTML element or elements that will be affected, followed by a series of **property** and **value** pairs (each pair is also called as **declaration**).



Declaration Blocks

The series of declarations is also called the **declaration block**.

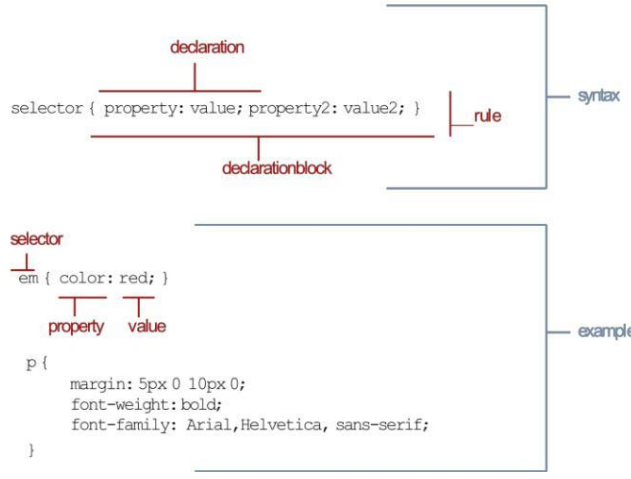
- A declaration block can be together on a single line, or spread across multiple lines.
- The browser ignores white space
- Each declaration is terminated with a semicolon.



Selectors

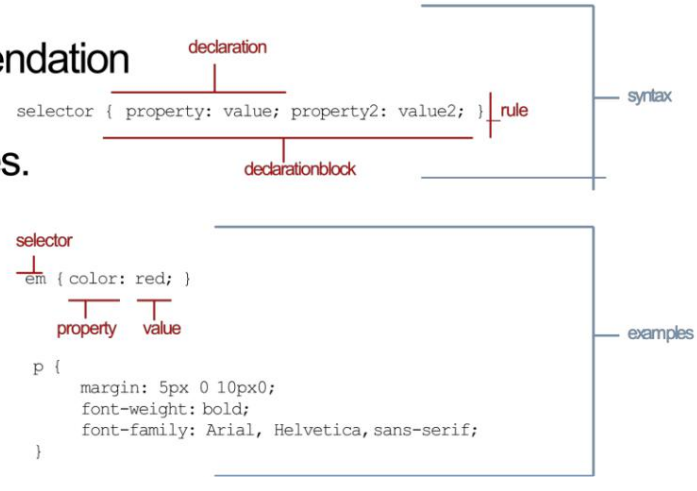
- Every CSS rule begins with a **selector**.
- The selector identifies which element or elements in the HTML document will be affected by the declarations in the rule.

Another way of thinking of selectors is that they are a pattern which is used by the browser to select the HTML elements that will receive the style.



Properties

- Each individual CSS declaration must contain a **property**.
- These property names are predefined by the CSS standard.
- The CSS 2.1 Recommendation defines over a hundred different property names.



Values

Each CSS declaration also contains a **value** for a property.

- The unit of any given value is dependent upon the property.
- Some property values are from a predefined list of keywords.
- Others are values such as length measurements, percentages, numbers without units, color values, and URLs.

LOCATION OF STYLES

- **Author-created style sheets** (what we are learning in this presentation).
- **User style sheets** allow the individual user to tell the browser to display pages using that individual's own custom a style sheet. This option is available in a browser usually in its accessibility options area.
- The **browser style sheet** defines the default styles the browser uses for each HTML element.

CSS style rules can be located in three different locations.

- Inline
- Embedded
- External

1) Inline Styles

```
<h1>Share Your Travels</h1>
<h2>style="font-size: 24pt"Description</h2>
...
<h2>style="font-size: 24pt; font-weight: bold;">Reviews</h2>
```

LISTING 3.1 Internal styles example

- An inline style only affects the element it is defined with in and will override any other style definitions for the properties used in the inline style.
- Using inline styles is generally discouraged since they increase bandwidth and decrease maintainability.
- Inline styles can however be handy for quickly testing out a style change.

2) Embedded Style Sheet

```
<head lang="en">
    <meta charset="utf-8">
    <title>Share Your Travels -- New York - Central Park</title>
    <style>
        h1 { font-size: 24pt; }
        h2 {
         font-size: 18pt;
         font-weight: bold;
        }
    </style>
</head>
<body>
    <h1>Share Your Travels</h1>
    <h2>New York - Central Park</h2>
    ...
```

LISTING 3.2 Embedded styles example

- While better than inline styles, using embedded styles is also by and large discouraged.
- Since each HTML document has its own <style> element, it is more difficult to consistently style multiple documents when using embedded styles.

3) External Style Sheet

```
<head lang="en">
   <meta charset="utf-8">
   <title>Share Your Travels -- New York - Central Park</title>
   <link rel="stylesheet" href="styles.css" />
</head>
```

LISTING 3.3 Referencing an external style sheet

This is by far the most common place to locate style rules because it provides the best maintainability.

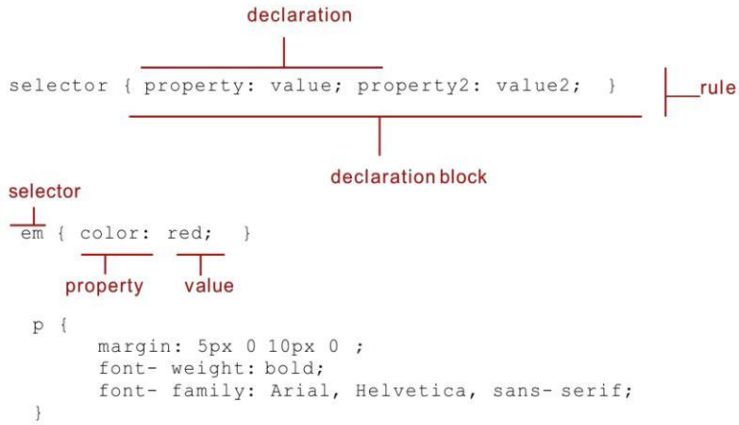
- When you make a change to an external style sheet, all HTML documents that reference that style sheet will automatically use the updated version.
- The browser is able to cache the external style sheet which can improve the performance of the site.

SELECTORS

1) Element Selectors

Usesthe HTMLelement name.

You can select al elements by using the **universal element selector**,which is the * (asterisk) character.



2) Grouped Selectors

```
/* commas allow you to group selectors */
p, div, aside {
   margin: 0;
   padding: 0;
}
/* the above single grouped selector is equivalent to the
   following: */
p {
   margin: 0;
   padding: 0;
}
div {
   margin: 0;
   padding: 0;
}
aside {
   margin: 0;
   padding: 0;
}
```

LISTING 3.4 Sample grouped selector

- You can select a group of elements by separating the different element names with commas.
- This is a sensible way to reduce the size and complexity of your CSS files, by combining multiple identical rules into a single rule.

Class Selectors

- A **class selector** allows you to simultaneously target different HTML elements regardless of their position in the document tree.
- If a series of HTML element have been labelled with ***the same class attribute value***, then you can target them for styling by using a class selector, which takes the form: period (.) followed by the class name.


```
<head>
<title>Share Your Travels </title>
<style>
.first {
font-style: italic; color: brown;
}
</style>
</head>
<body>
<h1 class="first">Reviews</h1>
<div>
<p class="first">By Ricardo on <time>September 15, 2012</time></p>
<p>Easy on the HDR buddy.</p>
</div>
<hr/>
<div>
<p class="first">By Susan on <time>October 1, 2012</time></p>
<p>I love Central Park.</p>
</div>
<hr/>
```

Id Selectors

- An **id selector** allows you to target a specific element by its id attribute regardless of its type or position.
- If an HTML element has been labelled with an id attribute, then you can target it for styling by using an id selector, which takes the form: pound/hash (#) followed by the id name.

```
<head lang="en">
<meta charset="utf-8">
<title>Share Your Travels -- New York - Central
Park</title> <style>
#latestComment {
font-style: italic; color: brown;
}
</style>
</head>
<body>
<h1>Reviews</h1>
<div id="latestComment">
<p>By Ricardo on <time>September 15,
2012</time></p> <p>Easy on the HDR buddy.</p>
</div>
<hr/>
<div>
<p>By Susan on <time>October 1,
2012</time></p> <p>I love Central Park.</p>
</div>
<hr/>
</body>
```


Attribute Selectors

- An **attribute selector** provides a way to select HTML elements by either the presence of an element attribute or by the value of an attribute.
- This can be a very powerful technique, but because of uneven support by some of the browsers, not all web authors have used them.
- Attribute selectors can be a very helpful technique in the styling of hyperlinks and images.

Pseudo Selectors

- A **pseudo-element selector** is a way to select something that does not exist explicitly as an element in the HTML document tree but which is still a recognizable selectable object.
- A **pseudo-class selector** does apply to an HTML element, but targets either a particular state or, in CSS3, a variety of family relationships.
- The most common use of this type of selectors is for targeting link states.

THE CASCADE: HOW STYLES INTERACT

- The "Cascade" in CSS refers to how conflicting rules are handled.
- The visual metaphor behind the term **cascade** is that of a mountain stream progressing downstream over rocks.
- The downward movement of water down a cascade is meant to be analogous to how a given style rule will continue to take precedence with child elements.

CSS uses the following cascade principles to help it deal with conflicts:

- **inheritance,**
- **specificity,**
- **location**

1) Inheritance

- Many (but not all) CSS properties affect not only themselves but their descendants as well.
- Font, color, list, and text properties are inheritable.
- Layout, sizing, border, background and spacing properties are not.

2) Specificity

- **Specificity** is how the browser determines which style rule takes precedence when more than one style rule could be applied to the same element.
- The more *specific* the selector, the more it takes precedence (i.e., overrides the previous definition).

Specificity Algorithm

- First count 1 if the declaration is from a 'style' attribute in the HTML, 0 otherwise (let that value =a).
- Count the number of ID attributes in the selector (let that value =b).
- Count the number of other attributes and pseudo-classes in the selector (let that value =c).
- Count the number of element names and pseudo-elements in the selector (let that value= d).
- Fina**l** y, concatenate the four numbers a+b+c+d together to calculate the selector's specificity.

Location

- When inheritance and specificity cannot determine style precedence, the principle of **location** will be used.
- The principle of location is that when rules have the same specificity, then the latest are given more weight.

The Box Model

- In CSS, a**ll** HTML elements exist within an **element box**.
- It is absolutely essential that you familiarize yourself with the terminology and relationship of the CSS properties within the element box.

Background Properties

Property	Description
background	A combined short-hand property that allows you to set the background values in one property. While you can omit properties with the short hand, do remember that any omitted properties will be set to their default value.
background-attachment	Specifies whether the background image scrolls with the document (default) or remains fixed. Possible values are: fixed, scroll.
background-color	Sets the background color of the element.
background-image	Specifies the background image (which is generally a jpeg, gif, or png file) for the element. Note that the URL is relative to the CSS file and not the HTML. CSS3introducedthe ability to specify multiple background images.
background-position	Specifies where on the element the background image will be placed. Some possible values include: bottom, center, left, and right. You can also supply a pixel or percentage numeric position value as well. When supplying a numeric value, you must supply a horizontal/vertical pair; this value indicates its distance from the top left corner of the element.
background-repeat	Determines whether the background image will be repeated. This is a common technique for creating a tiled background (it is in fact the default behaviour). Possible values are: repeat, repeat-x, repeat-y, and no-repeat.
background-size	New to CSS3,this property lets you modify the size of the background image.

Borders

- Borders provide a way to visually separate elements.
- You can put borders around all four sides of an element, or just one, two, or three of the sides.

Property	Description
border	A combined short-hand property that allows you to set the style, width, and color Of a border in one property. The order is important and must be: **border-style border-width border-color** Specifies the line type of the border. Possiblevalues are:solid,
border-style	dotted, dashed, double, groove, ridge, inset, and outset.
border-width	The width of the border in a unit(but not percents). A variety of keywords (thin, medium, etc) are also supported.
border-color	The color of the border in a color unit.
border-radius	The radius of a rounded corner.
border-image	The URL of an image to use as a border.

Margins

Did you notice that the space between paragraphs one and two and between two and three is the same as the space before paragraph one and after paragraph three?
This is due to the fact that adjoining vertical margins collapse.

Width and Height

- The width and height properties specify the size of the element's content area.
- Perhaps the only rival for collapsing margins in troubling our students, box dimensions have a number of potential issues.