Cascading Style Sheet (CSS):

Defining Style Sheets: A style sheet is usually a text file that is separate from your HTML document. Style sheets can also be internal, residing within your HTML code. A style sheet holds formatting codes that control your Web page's appearance. You can use style sheets to change the look of any Web page element, such as paragraphs, lists, backgrounds, and more. Anytime you want to apply the formatting from an external style sheet to an HTML document, you attach the style sheet to the page using a LINK tag. Style sheet files have a .css file extension.

Problems of HTML Formatting

1: One of the problems with using HTML for formatting is that it only offers a limited set of options to style your pages: You can use tags like <i>, <b>, and to change the appearance of text and use attributes like bgcolor to change the background color of HTML elements. You also have a number of attributes at your disposal for changing the way links appear in your page. Obviously, this feature set isn't rich enough to create the attractive web pages that your users expect and demand.

2: Another problem of HTML with a lot more impact on how you build your web pages is the way the styling information is applied to the page. By design, HTML forces you to embed your formatting in your HTML document, making it harder to reuse or change the design later. Consider the following example:

This is red text, in an Arial type face and slightly larger than the default text. The problem with this code snippet is that the actual data (the text in the element) is mixed with the presentation (the formatting of the text with the <font> tag in this example).

3: another problem with HTML formatting is the fact that you can't easily change the formatting at runtime in the user's browser: With the HTML from the previous code snippet, there is no way to let your visitor change things like the font size or color, a common request to help people who are visually impaired. If you want to offer your visitors an alternative version of the page with a larger font size or a different color, you'd need to create a copy of the original page and make the necessary changes.

4: The final problem with HTML formatting is that all additional markup in your page adds considerably to the size of the page: This makes it slower to download and display as the information needs to be downloaded with each page in your web site. It also makes it harder to maintain your pages as you'd need to scroll through large HTML files to find the content you need.

CSS allows you to overcome all of these problems.

How CSS Fixes Formatting Problems

CSS is designed to format your web pages in almost every possible way. It offers a rich set of options to change every little aspect of your web page, including fonts (size, color, family, and so on), colors and background colors, borders around HTML elements, positioning of elements in your page, and much more. CSS is widely understood by all major browsers today, so it's the language for visual presentation of web pages and very popular among web developers.

CSS overcomes the problem of mixed data and presentation by allowing you to define all formatting information in external files. Your ASPX or HTML pages can then reference these files and the browser will apply the correct styles for you. With this separation, the HTML document contains what you want to display, while the CSS file defines how you want to display it, enabling you to change or switch one of the two documents, leaving the other unmodified.

In addition, CSS can be placed directly in an HTML or ASPX page, which gives you a chance to add small snippets of CSS exactly where you need them. You should be cautious when placing CSS directly in an HTML or ASPX page, as you can then no longer control style information from a single, central location. Since all CSS code can be placed in a separate file, it's easy to offer the user a choice between different styles — for example, one with a larger font size. You can create a copy of the external style sheet, make the necessary changes, and then offer this alternative style sheet to the user.

Another benefit of a separate style sheet file is the decrease in bandwidth that is required for your site. Style sheets don't change with each request, so a browser saves a local copy of the style sheet the first time it downloads it. From then on, it uses this cached copy instead of requesting it from the server over and over again.

Controlling Multiple Pages

You can link every page in your Web site to a single style sheet. Any changes you make to the style sheet formatting are reflected in every HTML document linking to the sheet. By storing all the formatting information in one place, you can easily update the appearance of your site's pages in one fell swoop. This can be a real timesaver if your site consists of lots of pages.

Style Sheet Syntax

Style sheets are made up of rules, and each rule has two distinct parts: a selector and a declaration. The selector specifies the element to which you want to apply a style rule, and the declaration specifies the

formatting for the selector. For example, in the style rule H2 {color: silver}, the selector is H2 and {color: silver} is the declaration. When applied to a page, this rule will make all level 2 headings appear in silver. Figure 1 show how the elements are related to each other.

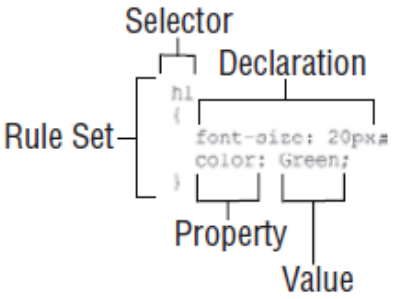

Figure 1: Cascading Style Sheet elements are related to each other

Style Sheet Declarations

A declaration consists of one or more property and value pairs such as font-size: 12px or position: absolute. The property and value are separated by a colon; multiple property-value pairs in a declaration are separated by semicolons. It is good form to put each property-value pair on a separate line when writing your rules. Similar to HTML, you can add extra spaces and line breaks to your style sheet code to make it more readable.

Inheritance

Tags you add inside other tags inherit the outer tag's formatting, unless you specify otherwise. For example, if you define a <BODY> style, any tags you nest within the <BODY> tags inherit that formatting. HTML inheritance makes it easy to keep the formatting consistent as you add new items within an element. The inheritance relationships between different styles in a document is known as the cascade.

External and Internal Style Sheets

You can connect an HTML document to an external or internal style sheet. Internal style sheets exist within an HTML page, between the <HEAD> and </HEAD> tags, while external style sheets are separate files. External style sheets are useful because you can link them to more than one HTML document. You might use an internal style sheet if your site consists of a single page.

Creating an External Style Sheet

You can use an external style sheet to define formatting and layout instructions and then apply those instructions to your HTML documents. You can save the style sheet as a text file and assign the .css file extension to identify the file as a Cascading Style Sheet.

Link to a Style Sheet:

You can link to a style sheet to assign a set of formatting rules to your HTML document. You can link multiple documents to the same style sheet to give all the pages in your site a consistent look and feel. within the <HEAD> and </HEAD> tags and add a new line. <LINK REL=”stylesheet” TYPE=”text/css” Type a blank space and HREF=”?”>, replacing ? with the name of the style sheet. If the style sheet is located in a subfolder, also specify the path. The style sheet is now linked with the page. You can test your page in a browser to see the style sheet results.

Create an Internal Style Sheet:

You can create an internal style sheet that resides within the <HEAD> tag of your HTML document. Internal style sheets are handy if your Web site consists of a single page because you can change both style rules and HTML in the same file.

Within the <HEAD> and </HEAD> tags, add a new line and type <STYLE>. Add a new line and type the element tag for which you want to create a style rule.

In this example, a new style rule is created for the H2 element.

```
H2
{
color: Red;
}
```

To be able to style an element on a page, a browser has to know three things:

- ❑ What element of the page must be styled?
- ❑ What part of that element must be styled?
- ❑ How do you want that part of the selected element to look?

The answers to these questions are given by selectors, properties, and values.

Selectors

As its name implies, a selector is used to select or point to a specific element within your page. A number of different selectors are available, giving you fine control over what element you want to style. The selector answers the first question: What element of the page must be styled?

The Universal Selector

The Universal selector, indicated by an asterisk (*) applies to all elements in your page. The Universal selector can be used to set global settings like a font family. The following rule set changes the font for all elements in your page to Arial:

```
*
{
font-family: Arial;
}
```

The Type Selector

The Type selector allows you to point to a specific HTML element. With a Type selector, all HTML elements of that type will be styled accordingly.

```
h1
{
color: Green;
}
```

This Type selector now applies to all <h1> tags in your code and gives them a green color. Type selectors are not case sensitive, so you can use both h1 and H1 to refer to the same heading.

The ID Selector

The ID selector is always prefixed by a hash symbol (#) and allows you to refer to a single element in the page. Within an HTML or ASPX page, you can give each element a unique ID using the id attribute. With the ID selector, you can change the behavior for that single element, like this:

```
#IntroText
{
font-style: italic;
}
```

Since you can reuse this ID across multiple pages in your site (it only has to be unique within a single page), you can use this rule to quickly change the appearance of an element that you use more than once, for example with the following HTML code:

I am italic because I have the right ID.

I am NOT italic because I have a different ID.

In this example, the #IntroText selector changes the font of the first paragraph — which has the matching id attribute — but leaves the other paragraph unmodified.

The Class Selector

The Class selector enables you to style multiple HTML elements through the class attribute. This is useful when you want to give the same type of formatting to a number of unrelated HTML elements. The following rule changes the text to bold for all HTML elements that have their class attributes set to Highlight:

```
.Highlight
{
font-weight: bold;
color: Red;
}
```

The following code snippet uses the Highlight class to make the contents of a element and a link (<a>) appear with a bold typeface:

```
This is normal text but <span class=”Highlight”>this is Red and Bold</span>
This is also normal text but
<a href=”CssDemo.html” class=”Highlight”>this link is Red and Bold as well</a>
```

Properties

Properties are the part of the element that you want to change with your style sheet. The CSS specification defines a long list of properties although you won’t use all of them in most web sites. The following table lists some of the most common CSS properties and describes where they are used.

Property	Description	Example
`display`	Changes the way elements are displayed, allowing you to hide or show them.	`display: none;` This causes the element to be hidden, and not take up any screen space.
`float`	Allows you to "float" an element in the page using a left or right float. Other content is then placed on the opposite side.	`float: left;` This setting causes other content following a float to be placed on the top right corner of the element. You'll see how this works later in the chapter.
`font-family` `font-size` `font-style` `font-weight`	Changes the appearance of fonts used on your page.	`font-family: Arial;` `font-size: 18px;` `font-style: italic;` `font-weight: bold;`
`height` `width`	Sets the height or width of elements in your page.	`height: 100px;` `width: 200px;`
`margin` `padding`	Sets the amount of free space inside (padding) and outside (margin) of an element.	`padding: 0;` `margin: 20px;`
`visibility`	Controls whether an element is visible in the page or not. Invisible elements still take up screen space; you just don't see them.	`visibility: hidden;` This causes the element to be invisible. However, it still takes up its original space in the page. It's as if the element is still there, but completely transparent.

Property	Description	Example
`background-color` `background-image`	Specifies the background color or image of an element.	`background-color: White;` `background-image: url(Image.jpg);`
`border`	Specifies the border of an element.	`border: 3px solid black;`
`color`	Changes the font color.	`color: Green;`

Fortunately, VWD helps you to find the right property with its many CSS tools, so you don't have to remember them all.

Values

Just as with properties, values come in many flavors. The values you have available depend on the property. For example, the color attribute takes values that represent a color. This can be a named color (such as White), or a hexadecimal number representing a red, green, and blue (RGB) component (such as #FF0000), or it can be set using the CSS RGB notation. The following examples are all functionally equivalent:

```
h1
{
color: Red;
}
h1
{
color: #FF0000;
}
h1
{
color: rgb(100%, 0%, 0%);
}
```

Apply a Style with the DIV Tag

You can apply styles to different sections of your Web page using the <DIV> tag. You can define style classes for the <DIV> tag in your external or internal style sheet, and then apply those classes in your HTML document. You can associate a class with <DIV> content using the CLASS attribute.