Web Module 4:

CSS, Part I

Introduction

In this module, the first of a two-module sequence on CSS (Part I and Part II), we'll introduce you to the joys of using CSS (Cascading Style Sheets) to determine the presentation (look, layout, formatting) of your web pages.

When you're done working through this module, you'll know:

- What CSS is and how it works
- How to create CSS style rules
- How to add CSS styles to a web page
- How to use CSS to style text and images
- How to use different style rule selectors

Caveat Lector

These eight Web modules build on each other. For example, in Module 4 it's assumed you understand the concepts and techniques from Modules 1-3. So, if you run into anything you're unfamiliar with in this module, browse through the earlier modules or turn to Google for help.

The Four Pillars of Web Design: A Refresher

In the HTML, Part I module we introduced the four pillars of web design:

I. Content – meaningful text, images, audio, and other media

II. Structure – the functional structure of the content, defined by HTML tags

III. Presentation – the appearance of the content, defined by CSS styles

IV. Behavior – the interaction of user and content, implemented with JavaScript or Flash

In the HTML, Part I and II modules we covered the first two pillars: adding content and structure to your web pages. In this module and its follow-up CSS, Part II module, we'll show you how to modify your web page presentation using CSS styles. Behavior is covered in the JavaScript and Forms modules.

Ready to learn how to use CSS to make beautiful web pages?

CSS Basics

What is CSS?

"A simple mechanism for adding style (e.g. fonts, colors, spacing) to Web documents." – W3C (World Wide Web Consortium)

"The preferred way (rather than HTML) to manage the presentation/look of your web pages." – rachMiel sZgut

What is the latest CSS standard?

CSS3.

CSS3 is completely backwards-compatible with earlier versions of CSS, so you don't have to worry about mixing features from different CSS standards.

What does CSS stand for?

Cascading Style Sheets.

What does "cascading" mean?

Multiple CSS style rules can be applied to the same HTML element in a web page: a <p> paragraph, for example. Conflicting rules sometimes compete for the same element. Cascading determines which rule takes precedence.

What is a style rule?

The specification of an individual CSS style. (More on this soon.)

Why should I use CSS (instead of HTML) for page presentation?

It enables the separation of structure (HTML) and presentation (CSS).

This is the one of the main guiding principles of web development: Keep structure separate from presentation.

HTML tags are used to structure a web page's content: organize it into chunks with specific structural meanings: paragraphs, lists, tables, divs, etc.

CSS styles are used to present a page's content: layout, font formatting, color schemes, line spacing, etc.

CSS provides way more exact control over presentation than HTML and is a much more efficient way of working on a website.

Edit one CSS stylesheet to change the look of an entire 100-page site. Cool, eh?

CSS can shorten HTML document length.

For example, multiple HTML font elements are unnecessary when you use CSS.

Creating CSS Style Rules

Here's the syntax of a CSS style rule:

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

selector – the HTML element to which the rule applies

property – the CSS property to be assigned a value

: – the assignment operator (equivalent to =)

value – the value assigned to the CSS property

; – the declaration ending character

A *property*: *value;* pair is called a declaration.

There can be multiple declarations in one style rule. This set of declarations is called a declaration block.

{ } – encloses all of the style rule's declarations (*property: value;* pairs).

Here's an example of a single-declaration style rule:

```
h2 { color: red; }
```

This style rule makes all h2 (level-2 heading) text in the page red. Its components:

selector – h2
property – color
value – red

And here's an example of a multi-declaration style rule

```
h2 {
color: red;
font-size: 22pt;
font-family: Arial;
}
```

Note that, as required, each declaration ends with a semicolon (;) to separate it from the next declaration, and all the declarations are enclosed in curly braces { } .

Note also that the layout (bracing, line breaks) of this style rule is different from that of the previous examples. More on this in a moment.

This style rule makes all h2 text in the page: red 22-point Arial. Its components:

selector – h2

property – color
value – red

property – font-size
value – 22pt (points)

property – font-family (font face)
value – Arial

Specifying Multiple Selectors in a Style Rule

As mentioned above, you can apply the same style rule to multiple HTML elements by specifying each element as a selector in the rule. Here's the syntax:

```
selector1, selector2, selector3, ... {
property: value;
}
```

The ... means that more selectors can follow.

For example, this style rule applies to four selectors, the h1 h2 h3 h4 headings:

```
h1, h2, h3, h4 { color: red; }
```

When you specify multiple selectors, make sure to separate each selector with a comma, and to leave the last selector comma-free.

Yes: `h1, h2, h3, h4 { color: red; }`

No: `h1, h2, h3, h4, { color: red; }`

Style Rule Layout

Different web developers use different layouts for coding their style rules.

> As long as the required components are present and the syntax is correct, all is well. Brace { } positioning and line breaks are left to the developers' discretion.

Here are some examples of popular style rule layouts. All of these style rules are identical from the browser's point of view, despite their differences in appearance:

```
h2 { color: red; font-size: 22pt; }

h2 { color: red; font-size: 22pt;
}

h2 {
color: red;
font-size: 22pt;
}

h2 {
   color: red;
   font-size: 22pt;
}

h2
{
color: red;
font-size: 22pt;
}

h2
   {
   color: red;
   font-size: 22pt;
   }
```

Tip: Since you'll be doing so much style-rule coding, and since consistency is such a key component of beautiful coding (see the HTML, Part II module), we recommend you come up with a style rule layout you love, and stick with through all your style rule coding.

Practice: Creating CSS Style Rules

Time to get your coding hands dirty and write some style rules. For now, don't worry about implementing the rules in a web page; we'll show you how to do that in the next section.

1. Write a style rule that displays all <p> (paragraph) text as 12-point Arial.
2. Modify the layout of your style rule (see above), with an eye to picking your favorite layout.
3. Staying with your favorite layout, write style rules to do the following:
   - Display all <blockquote> text as 9-point Times.
   - Display all <p> and <blockquote> text as orange.
   - Display all heading text – <h1> through <h6> – as green Calibri.

Adding CSS Styles to a Web Page

There are three ways to add CSS styles to your web pages: embedded, inline, and external.

Embedded styles

To create an embedded style, you insert the CSS style code into an HTML <style> element that resides in the <head> of the web page. Here's the syntax:

```
<style>
   selector { property: value; }
</style>
```

Here's an example of an embedded style that displays all <h1> and <h2> headers in the page in red 22-point characters:

```
<style>
h1, h2 {
color: red;
font-size: 22pt;
}
</style>
```

Embedded styles can have multiple style rules:

```
<style>
body { font-size: 22pt; }
p { text-align: center; }
h1, h2 { color: red; font-size: 22pt; }
</style>
```

Remember: The <style> element must reside in the <head> of the page, not the <body>.

```
<head>
<title>Page Title</title>
<style>
body { font-size: 22pt; }
p { text-align: center; }
h1, h2 { color: red; font-size: 22pt; }
</style>
</head>
```

Inline styles

To create an inline style, you use the HTML style attribute to insert the CSS style code into an opening HTML tag. Here's the syntax:

```
<htmlTag style="property: value;">styled content</htmlTag>
```

Note that the style code must be enclosed in straight – not curved! – quotes: " " .

For example, this inline style displays Welcome! as a green level-2 heading:

```
<h2 style="color: green;">Welcome!</h2>
```

Assigning multiple style declarations to a style attribute is permitted. Just make sure to end each declaration with a semicolon:

```
<h2 style="color: green; font-size: 24pt;">Welcome!</h2>
```

External styles

To create an external style, you insert the CSS style code into a separate file. This must be a plain-text file with a filename extension of .css as in: styles.css, mysite.css, etc.

HTML files are plain-text files with a .html extension; CSS files are plain-text files with a .css extension. JavaScript are plain-text files with a .js extension. See the pattern?

To link a web page to an external .css file – so that the style rules stored in this external file get applied to the HTML elements in the page – you add a <link> tag in the page's <head>:

```
<link href="URL" rel="stylesheet" type="text/css">
```

URL is a placeholder for the relative or absolute URL of your external .css file. If it were named styles.css and resided in a styles subfolder, you'd code:

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

The rel attribute specifies the relationship between the web page and the linked document (styles.css). Assigning "stylesheet" to rel tells the browser that the linked document is a CSS stylesheet to be imported.

You can only put CSS style rules in an external .css file, not <style> or any other HTML tags.

```
<style>
h1, h2 {
color: red;
font-size: 22 pt;
}
</style>
```

No: This .css file has HTML <style> tags.

```
h1, h2 {
color: red;
font-size: 22 pt;
}
```

Yes: Only CSS code goes in .css files.

When to use inline, embedded, and external styles

Web developers use inline styles for quick 'n dirty testing on single HTML elements.

For example, if you wanted to see what a paragraph would look like if the text were larger, adding an inline font-size style to the <p> tag would be easier than creating an embedded or external font-size style.

Web developers use embedded and external styles for their final CSS styling.

The advantage of using embedded styles is that the style code is in the same HTML file as the page content, so you don't have to manage two files. The disadvantage is that if you're building a site with consistent page appearance, you need to copy the style element into each page (HTML file) in the site. If you make a change to one embedded style, you need to copy that change into every page. And so on.

The (huge!) advantage of using external styles is that the style code is centralized in one .css file. If you make a change to a style in that file, it will automatically be applied to all the pages it's linked to. The disadvantage: You have to manage two files: HTML and CSS.

Our advice:

If you're doing a quick test on a single HTML element, use inline styles.

If you're a single page, use embedded or external styles, whichever you feel more comfortable with.

If you're creating a site with more than one page, store your styles in an external .css file. You don't want to have to paste the same style element into the head of 20 HTML files every time you change a style, do you?

Practice: Adding CSS Styles to a Web Page

1. Verify that all the style rules you wrote in the previous Practice task work as expected by embedded them in a <style> element in a web page and observing the results.
2. Move these style rules to an external .css file, add the necessary <link> element to the web page, and verify that they all work the same from the external file.
3. Use an inline style to "override" the external h2 style rule and display the text of a single <h2> element in the web page as black Calibri instead of green Calibri. (Note: All the other <h2> paragraphs on the page should still display in green.)

Using CSS to Style Text and Images

In this section, we'll show you how to use CSS to style a page's text and images.

You'll learn how to style links (<a> elements) in the CSS, Part II module.

For a complete list of CSS properties and property values, turn to ... you guessed it: the W3Schools CSS reference at w3schools.com/cssref/default.asp

Styling Text

Here's a list of the basic CSS properties for styling text along with examples of values that can be assigned to these properties and the actual CSS property:value declaration code.

CSS Text Styling Properties		
CSS Property	Examples of Values	Style Declarations

color
 	
Sets the text color.
	red
	#FF0000
	
	List of color names: w3schools.com/html/html_colornames.asp
		`color: red;`
		`color: #FF0000;`

font-family
 	
Sets the font face.
	Times
	Georgia
	Arial
	serif
	sans-serif
	monospace
		`font-family: Georgia, serif;`
		`font-family: Arial, sans-serif;`
		
		font-family lets you specify 2+ fonts. The first one (left to right) that is available in the browser is used. The last one is a catch-all for serif or sans-serif fonts.

font-size
 	
Sets the font size.
	14pt
	1em
	120%
		`font-size: 14pt;`
		`font-size: 1em;`
		`font-size: 120%;`

CSS Text Styling Properties (continued)		
CSS Property	Examples of Values	Style Declarations

font-weight
 	
Sets the font weight.
	bold
	bolder
	lighter
		`font-weight: bold;`
		`font-weight: bolder;`
		`font-weight: lighter;`

line-height
 	
Sets the line height.
	14pt
	14px
	140%
		`line-height: 14pt;`
		`line-height: 14px;`
		`line-height: 140%;`

text-align
 	
Sets text alignment.
	left
	right
	center
	justify
		`text-align: left;`
		`text-align: right;`
		`text-align: center;`
		`text-align: justify;`

Practice: Styling Text

1. Try out *all* of the above CSS text styling properties by coding style rules that include them.

   Feel free to embed your style rules in a <style> tag in the web page, or to store them in an external .css file – whatever way you find more convenient.

   While you're at it, try coding a couple of your text rules as inline styles.

Styling Images

Here's a list of the basic CSS properties for styling images along with examples of property values and CSS property:value declaration code.

Note: You use CSS to set an image's border, margins, and padding, not to modify its colors, brightness, contrast, etc. That's what Photoshop's for!

CSS Image Styling Properties		
CSS Property	Examples of Values	Style Declarations

border
 	
Displays a border around an image.
	1px
	solid
	black
		`border: 1px solid black;`
		
		Displays a 1-pixel thick solid black border around the image. You can customize your borders by changing any/all of these values.
		
		To display no border, either omit the border property, or use the declaration:
		
		`border: 0px;`

CSS Image Styling Properties (continued)		
CSS Property	Examples of Values	Style Declarations

padding
 	
Adds blank space inside the border (between border and image edges).
	20px;
	20pt;
	10%
		`padding: 20px;`
		`padding: 20px, 10px, 27px, 10px;`
		
		The first declaration adds 20 pixels of blank space between all four image edges and the border. The second adds 20 pixels between the top image edge and the border, 10 pixels between right edge and border, 27 pixels between bottom edge and border, and 10 pixels between left edge and border.
		
		The padding values go clockwise: top edge, right edge, bottom edge, right edge.

margin
 	
Adds blank space outside the border (between border and rest of page).
	20px;
	20pt;
	10%
		`margin: 20px;`
		`margin: 20px, 10px, 27px, 10px;`
		
		margin is like padding, except that padding adds blank space inside the border and margin adds it outside the border.

background-image Assigns to the <body> element an image that fills the entire page (like a desktop background).	url('image.jpg')	`background-image: url('image.jpg');` image.jpg is the URL of the background image. Note that it is enclosed in single quotes: ' ' .

Practice: Styling Images

1. Yep, you guessed it: Try out *all* these CSS image styling properties by coding them as embedded, external, or inline style rules.

Style Rule Selectors

As you'll recall from earlier in this module, a style rule selector is the HTML element to which the style rule applies. This rule applies a style to the <hr> element, making it red:

```
hr { color: red; )
```

There are three main types of style rule selectors:

- Element selector – assigns a style rule to all elements in the page with the element selector name
- Class selector – assigns a style rule to all elements in the page whose class attribute is set to the class selector name
- Id selector – assigns a style rule to the one (and only one!) element in the page whose id attribute is set to the id selector name

Let's have a look at each of these selector types.

The Element Selector

An element selector assigns a style rule to all the elements in the page with the same name as the element selector. Here's the syntax:

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

You use an element selector to style all instances of an HTML element such as <p> or <img>. In the following example, all <p> elements are displayed in a 24-point Georgia font:

```
p { font-family: Georgia; font-size: 24pt; }
```

As discussed above, you can assign a style rule to multiple selectors. In this example, all li (list item) and td (table cell) elements are displayed in a 12-point monospace font:

```
li, td { font-family: monospace; font: 12pt; }
```

You've already had practice coding CSS element selectors: All the style rules you've created up to now in this module have used element selectors.

The Class Selector

A class selector assigns a style rule to all the elements in the page whose class attribute is set to the class selector name. (Compare this with an id selector, which assigns a style rule to one single HTML element in a page.) Here's the syntax:

```
.class-selector-name { property: value; }
```

Class selector names always begin with a . (dot). For example:

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

To apply a class style to an HTML element, you add a class attribute to the element's opening tag and assign the class selector's name to it. Here's the syntax:

```
<HTML-element-name class="class-selector-name">
```

Do NOT include the beginning . (dot) of the class selector name in the class attribute value:

Yes: `<p class="makeRed">...</p>`
No: `<p class=".makeRed">...</p>`

Here's an example of the .makeRed class style in action. It is applied to two out of the total of five text elements on the page.

```
<head>
  <style type="text/css">
    .makeRed { color: red; }
  </style>
</head>

<body>
  <h3 class="makeRed">Due this week:</h3>
  <ol>
    <li>Poetry Reading</li>
    <li class="makeRed">Project 1</li>
  </ol>
  <h3>Due next week:</h3>
  <ol>
    <li>Project 2</li>
  </ol>
</body>
```

Practice: Using Class Selectors

1. Type the above example into a text editor, save it as an HTML file, and view it in a browser.
2. Modify the .makeRed style. Change the color, font size, and various properties of the <ol> (ordered list) element. Add the class="makeRed" attribute to other HTML elements.
3. Experiment!

The Id Selector

An id selector assigns a style rule to the one single HTML element in a page whose id attribute is set to the id selector name. (Compare this with a class selector, which can assign a style rule to multiple HTML elements in a page.) Here's the syntax:

```
#id-selector-name { property: value; }
```

id selector names always begin with a # (hashmark). For example:

```
#header { width: 100%; background-color: skyblue; }
```

To apply an id style to a HTML element, you add an id attribute to the element's opening tag. Here's the syntax:

```
<HTML-element-name id="id-selector-name">
```

Do NOT include the beginning # of the id selector name in the id attribute value:

Yes: `<div id="header">...</p>`

No: `<div id="#header">...</p>`

Here's an example of the #header id style in action. It is applied to just one element in the page: the <div> with its id set to "header". (You'll learn about <div> elements in the CSS II module.)

```
<head>
  <style type="text/css">
    #header {
    width: 100%;
    background-color: skyblue;
    font: bold 24pt/48pt Georgia;
    }
  </style>
</head>

<body>
  <div id="header">Page Header</div>
</body>
```

Practice: Using ID Selectors

1. Type the above example into a text editor, save it as an HTML file, and view it in a browser.
2. Make a simple page layout using CSS id styles and HTML <div> elements. Create a #main and a #footer id style, add two <div> elements to the page, assign one the id main and the other the id footer, and populate each with some text.
3. Experiment!

Selectors Galore!

There are many other kinds of style rule selectors.

Some selectors select one HTML element. Some select multiple elements.

In essence, selectors are *patterns* that select the HTML elements you want to style.

For a complete list of style rule selectors, turn to: w3schools.com/cssref/css_selectors.asp