TASK

Introduction to CSS

Visit our website

Introduction

WELCOME TO THE CSS AND THE BOX MODEL TASK!

You are now able to create basic web pages using HTML. You have likely by now wanted to be able to change the colour/font of text, or you may have wanted to change the size of an image or centre it. This has not been possible in terms of what you've learned so far, but in this task you will learn how to style your HTML elements. This task will introduce you to the basics of Cascading Style Sheets (CSS) and how to format various HTML elements using "in-line" and "internal" as well as "external" approaches. We'll also place close attention on the relevance of CSS selectors.

INTRODUCTION TO CSS

Cascading Style Sheets (CSS) is a language which is used to change the presentation and look of a particular document which has been written in a markup language, such as HTML. CSS is usually applied to web pages, but can also be used in other areas, such as XML documents.

INLINE STYLE

HTML **elements** are described using **attributes** and **properties**. You can style a web page by changing the properties of the elements that make up that webpage.

For example, any text that you add to a web page has several properties that you can change. These include the following properties:

- **font-family** - This specifies the font for the element (Arial, Times New Roman etc)
- **font-style** - This sets whether a font should be styled with a standard, *italic* or **oblique** face from its font-family.
- **font-size** - This sets the size of the font which can be specified as a fixed size in various units, a percentage, or as a predefined keyword.

An example of using the style attribute to change the font of an element is shown below:

```
<p style="font-family: 'Arial'; font-style: italic; font-size: 46px;">
      This is the paragraph where I describe myself.
```

```
</p>
```

Like all other attributes, the style attribute goes inside the element's beginning tag, right after the tag name. After specifying that you are changing the **style** attribute, you type **=**, and then, within double quotes, list the properties you want to change and after a colon, specify the value for that property:

<p style="font-family: Arial; font-style: italic; font-size: 46px;">

Attribute	Property	Value
style	**font-family**	**Arial**
	font-style	**Italic**
	font-size	**46px**

When you style an element individually by changing that element's properties, it is known as **inline styling**. Inline styling allows you to specify the style of an individual element in the line where that element is declared. What if you wanted to apply similar styles to all elements of a certain type, though? For example, what if you wanted to change the font of all paragraphs on your web page? You can do this by creating a CSS rule.

INTERNAL CSS

The example below shows how you can define a CSS rule in the **head** part of your HTML template. This is called **internal CSS**. The example below shows a CSS rule that will cause all paragraphs to be in the colour red and be of the font-family Arial. If the browser can't find Arial, then it will look for Helvetica. Paragraphs will also have a background colour of blue:

```
<head>
      <style>
            p {
                color: red;
                font-family: Arial, Helvetica;
                background-color: blue;
            }
      </style>
</head>
<p style="font-family: 'Arial'; font-style: italic; font-size: 46px;">
      This is the paragraph where I describe myself.
</p>
```

CSS follows the following syntax:

A style sheet consists of a selector and a declaration.

- The **selector** indicates which element you want to style. In the example above we are selecting the **p** element.
- The **declaration** block contains one or more declarations separated by semicolons. Examples of a declaration, as shown above, is:
    - **color: red;**
    - **font-family: Arial, Helvetica;**
    - **background-color: blue;**

  Declaration blocks should always be surrounded by curly braces.
- Each declaration includes a **property** and a **value**, separated by a colon.

See what other properties can be modified using CSS **here**.

You could use a combination of internal CSS (declared in the head of your HTML document) and inline style. How would the style rules apply? Essentially, the closer to the element the style is, the higher the precedence. For example, if you had the internal CSS rule shown in the code above in your page but you wanted the one paragraph to be styled differently from the rest, you would simply use inline style for that one paragraph and that would *overwrite* the rule specified by the internal CSS.

EXTERNAL CSS

If your website consists of many HTML files, you are likely to want to be able to apply the same style rules to all the web pages. To accomplish this you would need to use

external CSS instead of internal CSS. To do this, **create a separate file** with the extension .css. Within this file write all the style rules that you would like to specify. You can then link this external CSS file to all the HTML files in which you would like the style rules applied. To link an external CSS file (called examplesCSS.css in this example) to a specific HTML file, do the following:

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

In the <head> part of your HTML create a reference to your CSS file so that the styles can be used in your web page. Here, "href" refers to the name and path of your CSS file. In the example above, the file *exampleCSS.css* is in the same folder as the HTML page; "rel" says what sort of relation the file is to the HTML - i.e. the stylesheet; and finally, "type" tells the browser what sort of file it is and how to interpret it.

INTERNAL, EXTERNAL, OR INLINE: WHICH APPROACH IS THE BEST?

If we were to include the CSS shown in the image below in our CSS file, the result would be the same as if it were in the <style> tags in the HTML page. Is it better to use internal or external CSS? Generally, it is ***better to use external CSS wherever possible***. Why? *Readability* is an important factor. Imagine trying to read through a whole bunch of different languages at once (CSS, HTML, JavaScript) all in one file. Rather separate them – it's much easier to follow what's happening, especially when you are building a fancy website where plenty of different styles are being used.

```
p {
      color: red;
      font-family: Arial, Helvetica;
      background-color: blue;
}

body {
      text-align: center;
}
```

Another important reason to separate CSS from HTML files is to improve the *maintainability* of your website. If only external CSS is used for a website, you could update the look and feel of the website easily by simply replacing the external CSS file. Using external CSS also makes it easier to *debug* errors since all the CSS is in one place.

You may find, though, that it is necessary to use a combination of external, internal and inline style. In this case, it is important to understand the concept of *cascading*.

CSS VALIDATOR

As you have no doubt come to realise with HTML, as a developer it is very important to follow the syntax rules when developing websites. The same is true of CSS. You need to follow the rules for formatting your CSS rules exactly or unexpected errors will occur when you try to view your web page in the browser. Examples of common errors include spelling the name of an element incorrectly, not having matching opening and closing braces { }, or leaving out semicolons, ";", or colons, ":". You are bound to make mistakes that will violate these rules and that will cause problems when you try to view web pages in the browser - we all make syntax errors! Often! Being able to identify and correct these errors becomes easier with time and is an extremely important skill to develop.

To help you identify errors in your CSS, use this helpful **tool**.

Extra resource

The additional reading for this task includes two excellent resources:

1. The eBook entitled "HTML5 notes for professionals" by the 'beautiful people of Stack Overflow' that was included in the folder for your previous task (HTML).
2. **Web Style Sheets CSS tips & tricks: Centering things** by W3C.

CSS SELECTORS

A CSS selector attaches to the HTML elements on our page allowing for customised styling. Let's take a closer look at some of the most common CSS selectors you're likely to encounter and use.

Element selector

The element selector is the most basic type of CSS selector. It allows you to specify the precise HTML element you wish to style. This selector pinpoints an element tag and applies the same style to each element with that specific tag name. Notice that this

selector **does not have a unique name** – it is simply named according to its tag name.

```
<head>
   <style>
      p {
            text-align: center;
            color: blue;
      }
   </style>
</head>
<body>
  <p>This style will be applied on every paragraph.</p>
  <p id="para1">Me too!</p>
  <p>And me!</p>
</body>
```

ID selector

An **id** selector calls an HTML Element by its unique id name. It's unique because you can't give the same ID name to any other HTML Element in the same webpage. The id selector is really just an attribute inserted at the beginning of an html tag.

Notice that inside the double quotation, the value of **id** is given. Also notice that the **id** selector is called using hash (#), followed by id name (head1) and (para).

```
<style>
    #head1 { color: green }
    #para { color: red; background: yellow }
    p { color: blue }
</style>

<body>
    <h3 id="head1">Okay, let's make this h3 tag green.</h3>
    <p id="para">Now let's make use of our #para declarations.</p>
    <p>This paragraph does not have an id.</p>
</body>
```

The result will look something like this:

Okay, let's make this h3 tag green.
Now let's make use of our #para declarations.
This paragraph does not have an id.

Class selector

The class selector differs slightly from an id selector. Whereas an id selector pinpoints an individual HTML element, a **class** selector aims to change **all** HTML elements associated with that class. You'll also notice that a class selector begins with a '**.**' (dot).

```
<head>
   <style>
      p.right_align {
        text-align: right;
        color: blue;
        font-weight: bold;
      }
   </style>
</head>
<body>
<h1 class="right_align">This h1 text will not change because it does not contain the p tag.</h1>
<p class="right_align">This paragraph will change to blue, bold, and align to the right.</p>
</body>
```

The result will look something like this:

This h1 text will not change because it does not contain the p tag.

This paragraph will change to blue, bold, and align to the right.

Each selector gives us a greater degree of customisation. You may not see it just yet, but when you're working with multiple files and style elements, these handy selectors will save you that pain of manually adding styling for each line of code you write.

These selectors are the building blocks for more complex styling choices. Okay, let's have a look at the illustrations below for a clearer view of selectors and other CSS styling:

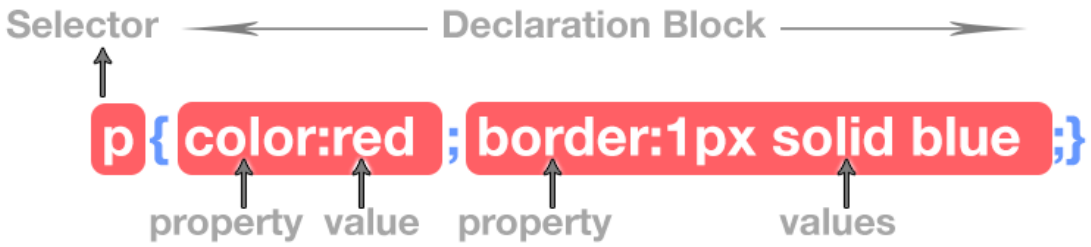

Image source: **https://tutorial.techaltum.com/cssselectors.html**

There's no need to memorise the image below, but you may find it helpful to learn to recognise the difference between a declaration, id, class, and HTML element:

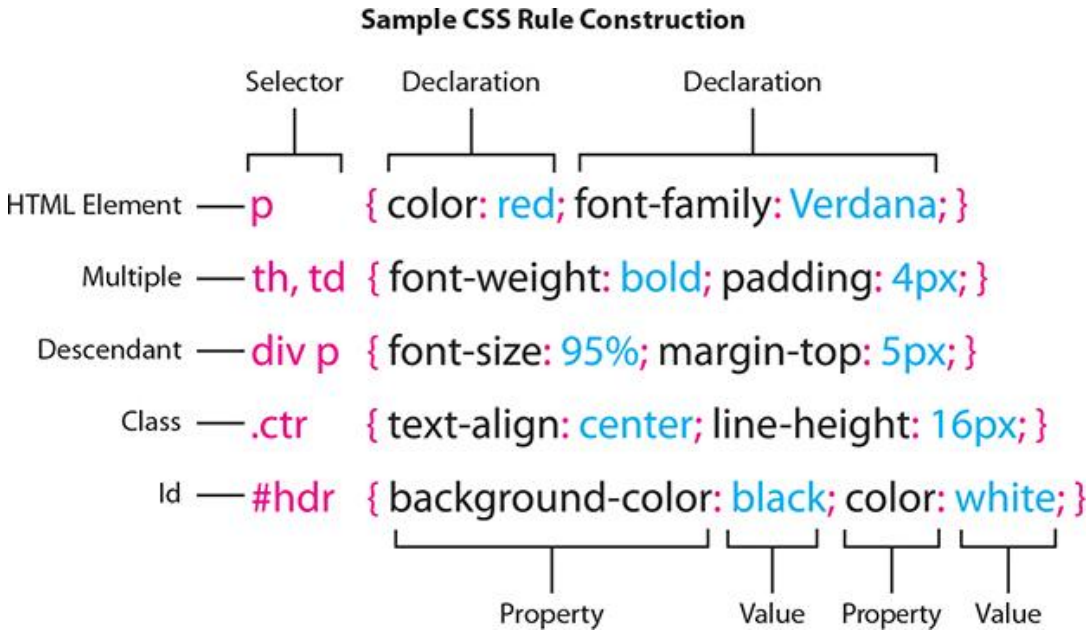

Image source: https://www.adobepress.com/articles/article.asp?p=2999389&seqNum=5

Instructions

Open all the examples in the directory called "Examples" for this task and read through the comments before attempting these tasks. Also, please consult the

additional reading. This additional reading is a resource provided by the World Wide Web Consortium (W3C). The W3C is the international community that develops the standards that govern the web. In other words, these are the folks that make the rules for how the web works!

A note from the
HyperionDev Team

Please have a look at the following video playlist which explains CSS, **here**.

Compulsory Task

In this task, you will need to create a tribute page. This web page will be about someone you admire in your life. You can find an example of a tribute page **here**.

- Create files called **tribute.html** and **myStyles.css** in this folder. The **tribute.html** file should include:
    - a title or a heading
    - an image
    - a caption for the image
    - a timeline of the life of the tribute in the form of a list.
- Set out the basic HTML document template and title it as you see fit.

- Use the following eight elements at least once:
    - h1
    - p
    - img
    - h2
    - h3
    - nav (find help **here** and **here**.)
    - header
    - aside
- Depending on the content you have chosen, align some on the left, some on the right, and some in the middle.
- The nav bar should be centred and the elements placed next to each other (horizontally).
- Change the background colour of the entire page (hint: think about where all of the content is kept).
- Make all the headings appear in italics, and change their font family to Times New Roman.
- The following should be added to your elements (1 per element)
    - Text-colour
    - Image-size
    - Border
    - Padding
- Before submitting your code, check it with the HTML and CSS validators located **here** and **here**.

Rate us

Share your thoughts

HyperionDev strives to provide internationally-excellent course content that helps you achieve your learning outcomes.

Think that the content of this task, or this course as a whole, can be improved, or think we've done a good job?

Click here to share your thoughts anonymously.

REFERENCES

CSS reference - CSS: Cascading Style Sheets | MDN. Mozilla.org. Published June 16, 2022. Accessed June 28, 2022. **https://developer.mozilla.org/en-US/docs/Web/CSS/Reference**

Powell K. HTML & CSS for beginners. YouTube. Published online 2022. Accessed June 28, 2022. **https://www.youtube.com/playlist?list=PL4-IK0AVhVjM0xE0K2uZRvsM7LkIhsPT-**

Pinimg.com. Published 2022. Accessed June 29, 2022. **https://i.pinimg.com/originals/ab/6b/61/ab6b61d2f4197fb41a07fa04038836df.png**