Hyperiondev

TASK

CSS Overview

Visit our website

Introduction

WELCOME TO THE CSS OVERVIEW TASK!

In the previous task you added HTML elements to a webpage. Therefore, your webpage contains the information you want but it probably doesn't look that great! Don't worry. In this task, you will learn to use CSS to position and style the elements that you have added to your webpage to make it much more attractive and exciting!

INTRODUCTION TO CSS

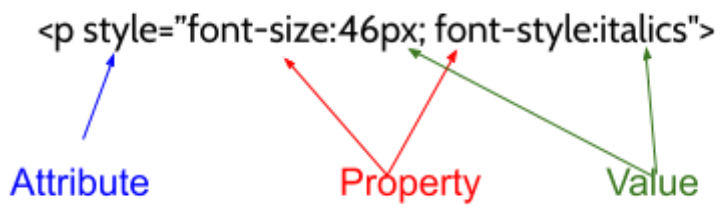
Cascading Style Sheets (CSS) is a language that is used to change the presentation and look of a particular document that has been written in a markup language, such as HTML. CSS is usually applied to web pages, but can also be used in other areas, such as XML documents.

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. For example: font-family (Arial, Times New Roman etc), font-style (normal, italics etc) and font-size. An example of using the style attribute to change the font of an element is shown below:

```
<p style="font-size:46px;color:blue;">
      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.



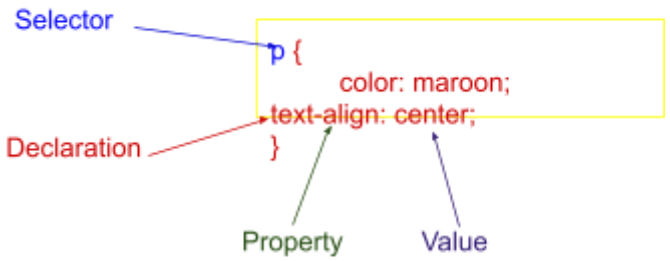
When you style an element individually by changing that element's properties, it is known as **inline styling**. Inline styling allows you to specify the style of an individual element in the line where that element is declared. What if you wanted to apply similar styles to all elements of a certain type? 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!

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

CSS syntax consists of a selector and a declaration.



- The **selector** indicates which HTML element you want to style.
- The **declaration** block contains one or more declarations separated by semicolons. A declaration always ends with a semicolon and is surrounded by curly braces.
- Each declaration includes a **property** and a **value**, separated by a colon.

Extra resource

Explore other **CSS properties** that can be modified.

The CSS below should look familiar. Here the selector is always an *element*.

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

However, you can also use *class* and *ID selectors*. A class selector is used when the selector describes the rule for all elements that have a *class attribute with the same name defined*. An ID selector describes the style of an element with a specific ID attribute defined.

Open **example.html**. Notice that in this file we have several elements for which either the class or ID attribute has been defined. For example, notice how there are several **<a>** elements that have the class attribute set to "**moreButton**" as shown below.

```
<a href="" class="morebutton">more</a>
```

However, there are also **<a>** elements that do not have a class attribute specified.

```
<li><a href="contact.html">contact</a></li>
```

Therefore, instead of creating a CSS with an element selector ('**a**'), we could create a rule that is specific to a class selector. See an example of this below.

```
.moreButton {
        font-weight: bold;
    }
```

When you use a class selector, the name of the class will always be preceded by a dot, ‘ **.** ’. The style rule will cause all elements where the class attribute has been set to **class="moreButton"** to have bold text.

Similarly, you could create a stylesheet that uses ID selectors. ID selectors start with a hash, ‘**#**’, as is shown in the example below.

```
#firstmorebutton {
        font-weight: bolder;
        font-family: cursive;
    }
```

The rule above would cause the text of the element where the ID attribute equals “**firstmorebutton**” to be bold and cursive.

Although you can have many elements that have a class attribute with the same value, an ID name must be unique in the document!

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 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, 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 to a specific HTML file, do the following:

```
<link href = "examplesCSS.css" rel = "stylesteet" 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. “**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. "`type`" tells the browser what type 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 below in our CSS file, the result would be the same as if it were in the `<style>` tags in the HTML page. So 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, Python) all in one file. Rather separate them — it's much easier to follow what's happening, especially when you are building fancy websites 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 you wanted to update the look and feel of a website, this could easily be done by simply replacing the external CSS file if only external CSS is used for the website. 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 cascade.

Take note:

Here are two more cool CSS tricks:

1. If you would like to apply a certain style rule to an element when it is in a certain state (e.g. if you hover over it) you can do this as shown below:

```
/* Any button over which the user's pointer is hovering */
button:hover {
  color: blue;
}
```

In this example, 'hover' is a pseudo-class (a keyword that describes the state of the selector). Explore a **list of pseudo-classes.**

2. If you would like to apply a certain rule to an element that is a descendant (child) of another element, you can do so as shown below:

```
li li {
  list-style-type: circle;
}
```

The rule above will make all list items that are descendants of other list items have a circle as a bullet point. Learn more about using a **descendant combinator**.

USING CSS GRIDS FOR LAYOUT

Extra resource

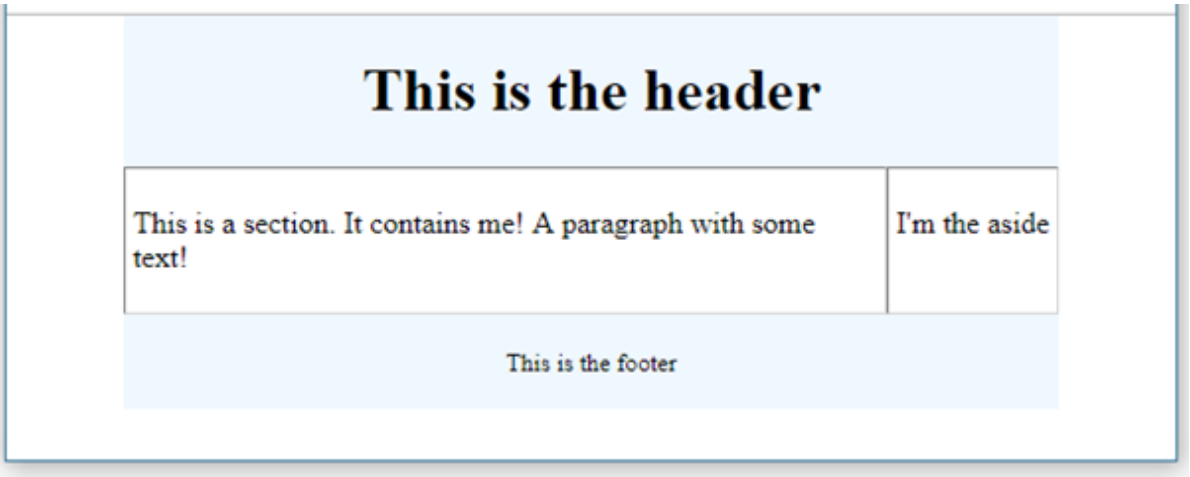
`<header>`, `<footer>`, `<aside>`, etc. elements are new sectioning elements used with HTML5. HTML5 is the latest evolution of the standard that defines HTML. Learn more about **HTML5** and find out more about the **`<article>` element** and how it compares to `<div>` elements..

Getting the layout of elements correct is one of the trickiest aspects of CSS. You will be required to do some research in this regard to complete this task successfully. Feel free to use whatever resources you like, but **Introduction to CSS layout** and **CSS layout cookbook** are excellent places to find help.

Assume that we have created the following HTML:

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <title>Position example</title>
    <link rel="stylesheet" href="example_style.css" />
</head>
<body>
    <div id="container">
        <header><h1>This is the header</h1></header>
        <section>
            <p>This is a section. It contains me! A paragraph with some text! </p>
        </section>
        <aside><p>I'm the aside</p></aside>
        <footer><p><small>This is the footer</small></p></footer>
    </div>
</body>
</html>
```

And we want this HTML to be displayed in the browser as shown in the image below:



It would be difficult to get this layout using only relative, absolute, static positioning. Instead, we can use a CSS grid template to achieve this layout.

A CSS grid is like a table that is designed to make it easier to position elements on a webpage. The grid usually contains 12 columns.

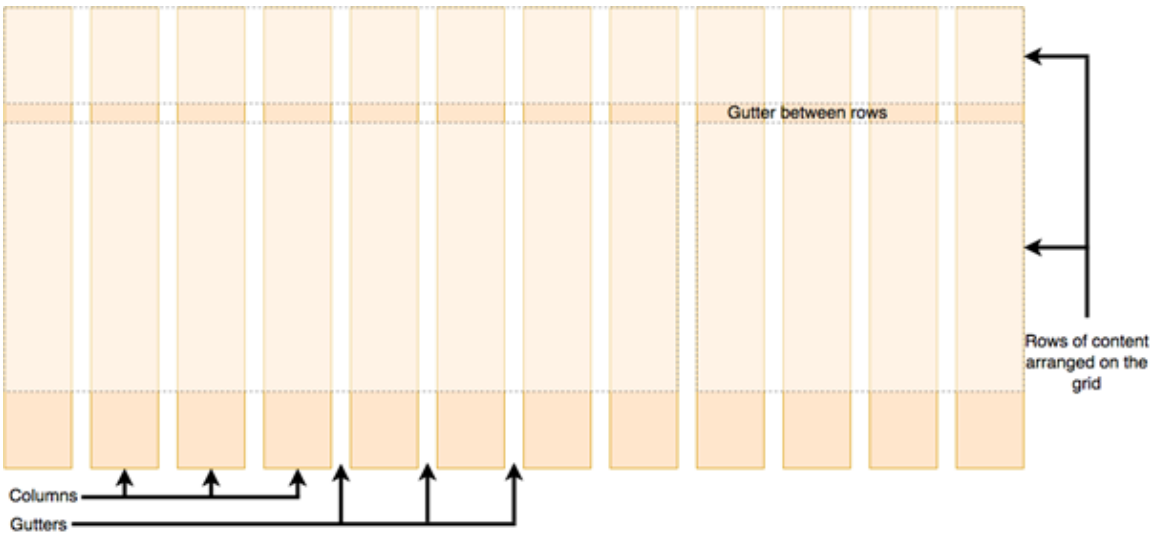


Image source:
https://developer.mozilla.org/en-US/docs/Learn/CSS/CSS_layout/Grids#A_CSS_Grid_grid_framework

The position of an element is described in terms of which row it is in and how many columns it takes up. To illustrate, look at the CSS rules below.

```
body {
    width: 80%;
    margin:auto;
}
#container {
    display:grid;
}
header {
    grid-column: 1/13;
    grid-row: 1;
    background-color: aliceblue;
    text-align: center;
}
section{
    grid-column:1/9;
    grid-row:2;
    padding:4px;
    border: 1px inset lightgrey;
}
aside {
    grid-column: 9/13;
    grid-row: 2;
    padding: 4px;
    border: 1px inset lightgrey;
}
footer{
    grid-column: 1/13;
    grid-row: 3;
    background-color:aliceblue;
    text-align: center;
}
```

We want the `<header>` element to span across the full width of the whole container/grid. Thus, we specify that we want it to start at the first line (1) of the grid and end at the last line (13) of the grid. To put the `<header>` element to be at the top of the grid, put it in the first row of the grid.

Place the `<section>` element in the second row of the grid. The element is eight columns wide, so the section element starts at the first line of the grid and ends at the 9th line.

CASCADE

As we know, CSS stands for cascading style sheets. You may have wondered why they are called *cascading* style sheets. Cascading has to do with how the rules are applied.

If your website contains external, internal and inline CSS, inline CSS overrides internal CSS rules and external CSS files. Internal CSS overrides external CSS rules. If there are conflicting rules regarding properties, properties override other properties, but entire rules don't override other rules. When several CSS rules match the same element, they are all applied to that element. Only after that are any conflicting properties evaluated to see which individual styles will win over others.

Another important rule to remember is that *the more specific a rule is, the higher its precedence*. For example, in a stylesheet that uses element selectors, class selectors and ID selectors, *element selectors are the least specific* (because they could match the most elements in a page) whereas ID selectors are the most specific. Therefore, ID selectors will be applied over class selectors and element selectors.

CSS VALIDATOR

As you have no doubt come to realise, as a developer it is very important to follow the syntax rules. 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

Remember that with our courses, you're not alone! To become a competent software developer, it is important to know where to get help when you get stuck.

Here you can find resources that provide extra information about **CSS**.

The additional reading for this task includes two excellent resources:

1. An eBook entitled "HTML5 notes for professionals" by the 'beautiful people of Stack Overflow' in the additional reading folder.
2. **Web Style Sheets CSS tips & tricks: Centering things** by W3C.

Compulsory Task

Follow these steps:

- Use CSS to style and position the elements on your webpage that you created in the previous task (**index.html**) as attractively as possible.

Completed the task(s)?

Ask an expert code reviewer to review your work!

Review work

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.