



TASK

CSS Primer

[Visit our website](#)

Introduction

WELCOME TO THE CSS OVERVIEW TASK!

In the previous task, you added HTML elements to a web page, but did not learn how to make the page look good to a viewer. 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! You will also learn how to create responsive and flexible web designs using the grid and flexbox layout.

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 to format XML documents.

In this task, you will go beyond a surface-level breakdown of CSS and its importance in the world of web development, and you'll get to explore one of the most popular programming frameworks, called Bootstrap. The good news is that you don't have to be a programming guru to use Bootstrap!

The deeper we go into a concept, the more complex we start to realise it is. CSS is no different. Thankfully, due to how popular programming is, we have other developers (in this case, the people that created Bootstrap are a great example!) to help us solve some of these complex issues and sometimes we don't even need to know the developers for us to get help!

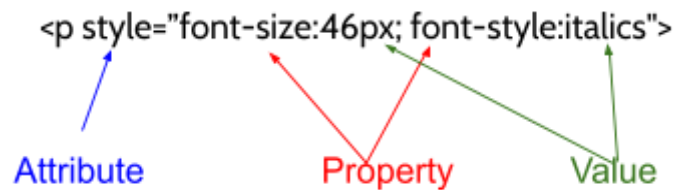
INLINE STYLE

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 web page. For example, any text that you add to a web page has several properties that you can change, including font family (Arial, Times New Roman, etc.), font style (bold, 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.

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 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 style sheet 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.

Note: Although you can have many elements that have a class attribute with the same value, each **ID** name must be unique within 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 instance, 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 `.css` extension. 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="exampleCSS.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 style sheet. “`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. 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 different languages (CSS, HTML, Python) simultaneously in one file. Rather separate them – it’s much easier to follow what’s happening, especially when building fancy websites with plenty of different styles.

```
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 your website and you wanted to update the site’s look and feel, this could easily be done 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 styles. In this case, it is important to understand the concept of cascade.



Extra resource

When creating websites, remember that you have the flexibility to design the web pages using a wide range of CSS properties. You can customise the appearance of the font, adjust the dimensions of images, or even add interactive hover effects when moving the mouse over HTML elements.

Here are two 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](#).

Note: CSS enhances the visual appearance and usability of the web page that improves user experience. Therefore, it is good practice to always attractively style your web pages to keep users engaged. Here are useful resources to further understand [CSS basics](#) and help you to apply various [CSS properties](#) when styling HTML elements.

RESPONSIVE AND FLEXIBLE DESIGN LAYOUTS

Now that you have understood how to use the different methods of applying CSS styles to HTML elements, let's explore how we can arrange and lay out elements on the web page using flexible and responsive layouts. Responsive design is a method

of creating a web application that is able to adapt to different screen resolutions while maintaining interactivity. In other words, the exact layout will vary, but the relationship between elements and the reading order remains the same. Here are two flexible designs to ensure that the layout remains consistent:

GRID LAYOUT

A CSS grid is like a table that is designed to make it easier to position elements on a web page. The grid usually contains 12 columns. The position of an element is described in terms of which row it is in and how many columns it takes up.

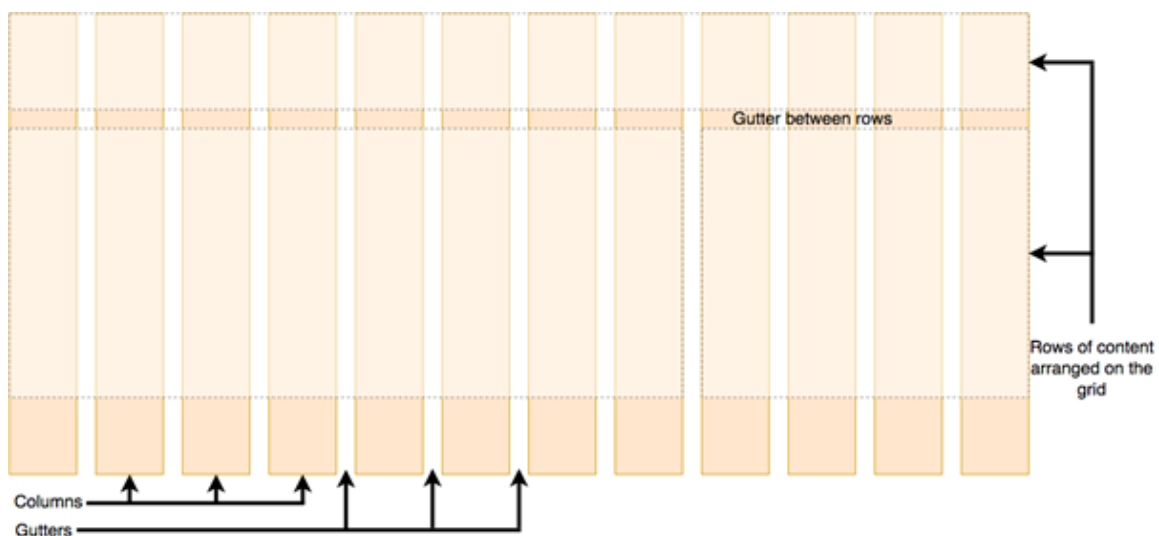


Image source: (2023). *Grids* [Illustration]. MDM Web Docs.

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

The grid layout is a powerful tool for creating fluid grids. A fluid grid breaks down the width of the page into several equally sized and spaced columns. You can then position the page content according to these columns. Each fluid column expands accordingly when the viewport expands horizontally, as does the content within the columns.

In the example below, we have defined a grid layout using the following properties:

- **Display:** To define a grid, we use the grid value of the display property. In the code below, the parent element, which has a class of “**grid-container**”, will define rows and columns within a container. The children elements, which have a class of “**grid-items**”, will occupy the cells within this container.
- **Grid-template-columns:** This property is used to specify that there are three equal columns in the grid. Note: **fr** represents a fraction of space taken up by each column.


```

<!DOCTYPE html>
<html>
  <head>
    <title>Grid Example</title>
    <style>
      .grid-container {
        display: grid;
        grid-template-columns: repeat(3, 1fr);
        background-color: purple;
      }
      .grid-item {
        background-color: plum;
        text-align: center;
        border: 1px solid black;
        padding: 10vh;
        font-size: 3vh;
      }
    </style>
  </head>
  <body>
    <div class="grid-container">
      <div class="grid-item">1</div>
      <div class="grid-item">2</div>
      <div class="grid-item">3</div>
      <div class="grid-item">4</div>
      <div class="grid-item">5</div>
      <div class="grid-item">6</div>
      <div class="grid-item">7</div>
    </div>
  </body>
</html>

```

FLEXBOX LAYOUT

Flexbox is a CSS module designed to more efficiently position multiple elements, even when the size of the contents inside the container is unknown. Items in a flex container expand or shrink to fit the available space.

Unlike grid layouts which use columns, a flexbox uses a single-direction layout to fill the container. A flex container expands items to fill the available free space or shrinks them to prevent overflow.

In the example below, we have defined a flexbox grid using the following properties:

- **Display:** To define a flexbox, the “**flex**” value of the display property is used. In the code below, the parent element, which has a class of “**flex-container**”, will contain flexible children elements. The children elements, which have a class of “**flex-item**”, will be aligned in one direction.
- **Flex-flow:** The **flex-flow** property is a combination of the **flex-direction** and **flex-wrap** properties, as it allows us to specify direction and wrapping. The **flex-direction** property establishes the main axis/direction by row or column, whereas the **flex-wrap** property allows items to wrap as needed, which is useful as by default flex items will all try to fit on one line.
- **Flex:** We add the rule “**flex: 1**” to our flex item. The flex property is a unitless proportion value that dictates how much space each flex item will take up along the central axis compared to other flex items. In this case, we're giving each “.**flex-item**” element the same value of one, which means they'll all take up an equal amount of the spare space left after properties like padding and margin have been set. An element with a flex value of two will take up twice as much of the available space.

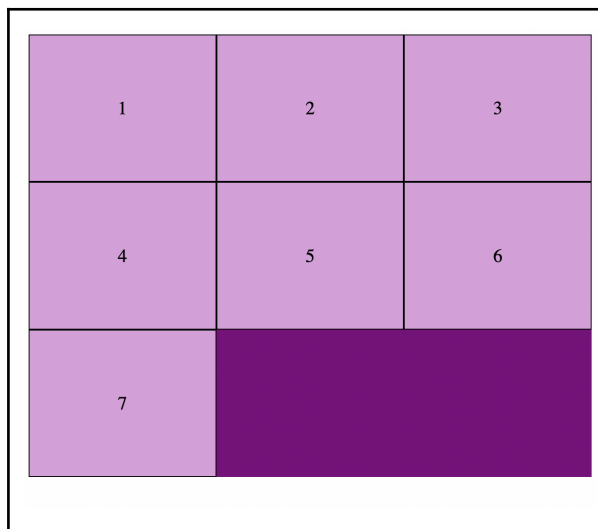
```
<!DOCTYPE html>
<html>
  <head>
    <title>FlexBox Example</title>
    <style>
      .flex-container {
        display: flex;
        flex-flow: row wrap;
        background-color: purple;
      }
      .flex-item {
        background-color: plum;
        text-align: center;
        border: 1px solid black;
        padding: 10vh;
        font-size: 3vh;
        flex: 1;
      }
    </style>
  </head>
  <body>
    <div class="flex-container">
      <div class="flex-item">Item 1</div>
      <div class="flex-item">Item 2</div>
      <div class="flex-item">Item 3</div>
      <div class="flex-item">Item 4</div>
      <div class="flex-item">Item 5</div>
      <div class="flex-item">Item 6</div>
      <div class="flex-item">Item 7</div>
      <div class="flex-item">Item 8</div>
      <div class="flex-item">Item 9</div>
      <div class="flex-item">Item 10</div>
    </div>
  </body>
</html>
```

```

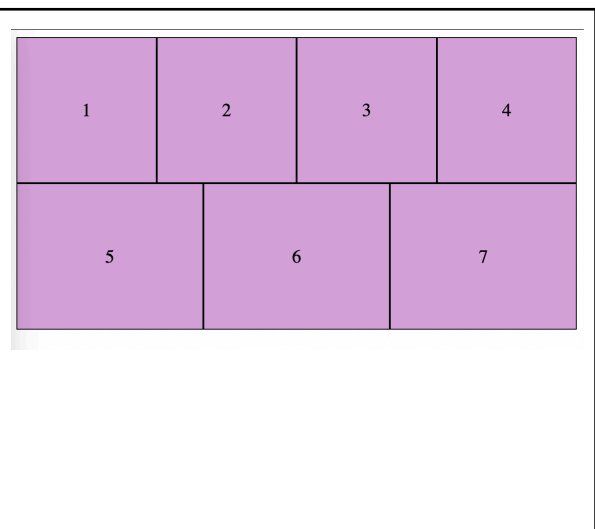
    </style>
</head>
<body>
  <div class="flex-container">
    <div class="flex-item">1</div>
    <div class="flex-item">2</div>
    <div class="flex-item">3</div>
    <div class="flex-item">4</div>
    <div class="flex-item">5</div>
    <div class="flex-item">6</div>
    <div class="flex-item">7</div>
  </div>
</body>
</html>

```

Grid container



Flex container



Let us now observe how each of these layouts are displayed in comparison to one another. The image on the left is a grid container made up of seven grid items. We can see that the items are aligned with the three columns that have been declared. Take note of the available grid space, represented by a darker purple.

The image on the right is a flex container made of seven items. All items in a row share the available space equally and succeed in filling the box regardless of the screen dimensions.



Extra resource

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 the task successfully. Feel free to use any resources you like, but [Introduction to CSS layout](#) and the [CSS layout cookbook](#) are excellent places to find help.

CASCADE

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 style sheet 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, it is very important to follow syntax rules. Nowhere is this more applicable than when using CSS. You need to follow the rules for formatting your CSS 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](#).

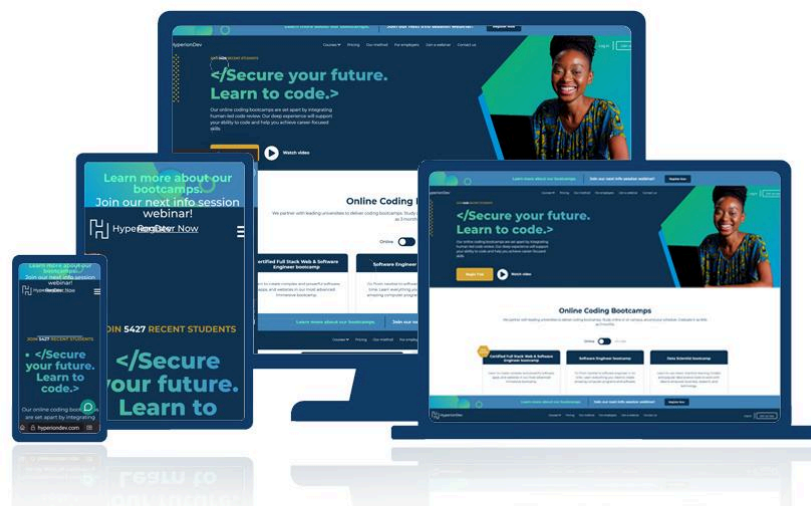


Extra resource

The additional reading for this task includes two excellent resources:

- An eBook entitled ***CSS notes for professionals***
- **Web Style Sheets CSS tips & tricks: Centering things** by W3C.

RESPONSIVE DESIGN



Take a look at the above picture. Notice how, in the world we live in, we have so many different screen sizes. Think about one of the most popular applications that people use nowadays, like WhatsApp. Have you ever noticed that regardless of the device that you use WhatsApp on, the screen always remains the same? This is one of the critical features of responsive design.

At this point in time, you most likely have only created one website design, and that design works with your laptop screen, but now imagine you had to look at your website on your mobile device or on a 4k TV screen.

As you can imagine, your website would end up breaking on those devices; this is because you haven't implemented any form of **responsive design**.

By now, you might be wondering what precisely responsive design is.

Well, responsive design is a web developer's way of creating a website that changes based on the size of the screen the user is using. Now, even though this takes a bit of time, you have Bootstrap to help you along the way. This is where Bootstrap comes into play!

WHAT IS BOOTSTRAP?

We've mentioned that there are many developers around the world who are creating systems for developers to use to help them save time and energy.

Bootstrap is one of the tools that have been created by developers to help other developers save time and effort during the process of creating systems.

It is known as a **framework**, which you will slowly become more familiar with as you progress through the course. Simply put, a framework is code written by someone else that you can use in your program to make your development progress quicker, and make your code more manageable and cleaner to write.

In this task, we'll be going through how to install Bootstrap, and how to use it to make your website responsive and well-structured!

INSTALLING BOOTSTRAP

Let's go through the quick process of "installing" Bootstrap (it's just like importing a CSS file into your HTML file!)

Just follow these steps:

1. Head over to [this link](#).
2. Scroll down the page till you see a heading that says "Include via CDN".
3. Copy the "CSS only" version – it will look something like this `<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.0/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-gH2yIJqKdNHPEq0n4Mqa/HGKIhSkIHeL5AyhkYV8i59U5AR6csBvApHHNL/vI1Bx" crossorigin="anonymous">` – note that this is not the whole url, just the start to help you find it.
4. Head into your HTML file:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <!-- CSS only -->
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.0/dist/css/bootstrap.min.css" rel="stylesheet"
    integrity="sha384-gH2yIJqKdNHPEq0n4Mqa/HGKIhSkIHeL5AyhkYV8i59U5AR6csBvApHHNL/vI1Bx" crossorigin="anonymous">
  <title>iTunes App</title>
</head>
<body>
```

a.

5. Paste the link in your head tags.

And that's it! You've successfully installed Bootstrap.

USING BOOTSTRAP

Just like any concept in programming, you will need to learn how to implement Bootstrap.

Before we can begin implementing it, we need to understand the backbone of Bootstrap and how it will calculate how to create a responsive website.

THE GRID SYSTEM

Whenever you look at a website, you're actually looking at a grid system! Take a look at the image below of our website.



As you can see, there are a lot of different colours on the screen! These are known as “columns” (usually invisible); this is what helps a web page identify where certain elements are. Bootstrap works with the grid system!

You may notice that there are 12 purple columns. This is the **maximum** number of columns Bootstrap can work with! This means that once you start designing your web pages using Bootstrap, you should never forget that you only have 12 columns to design your web page on. Work with the limit of 12 columns in the back of your head.

Take a look at the image below:

12											
6						6					
3			3			3			3		
1	1	1	1	1	1	1	1	1	1	1	1

This is an example of a grid pyramid that was created using Bootstrap and standard HTML. As you can see, we're now able to place some aspects next to each other using the different columns Bootstrap provides for us.

Because we're now making use of grids, what you'll find is that no matter how you resize the browser, the website will still stay on the page and automatically adjust itself to fit all the content in the way it was placed.

This can be done in minutes with Bootstrap, compared to writing our own version of the grid system using normal CSS!

Screen sizes

While Bootstrap is powerful, it becomes even more powerful once we start learning and implementing screen sizes. Bootstrap allows us to select what data will be displayed based on the screen size we provide.

Take a look at the screenshot below:

Class prefix	Screen Size	Grid behaviour	Container width	Suitable for
.col-	Extra small (<576px)	Horizontal at all times	None (auto)	Portrait phones
.col-sm-	Small (>=576px)	Collapsed to start, horizontal above breakpoints	540px	Landscape phones
.col-md-	Medium (>=768px)	Collapsed to start, horizontal above breakpoints	720px	Tablets
.col-lg-	Large (>=992px)	Collapsed to start, horizontal above breakpoints	960px	Laptops
.col-xl-	Extra Large (>=1200px)	Collapsed to start, horizontal above breakpoints	1140px	Laptops and Desktops

These are all the screen sizes that are used in Bootstrap. For the most part, when you work with Bootstrap, you will most commonly use "col-md" (medium device) as you are most likely working on a tablet/laptop.

However, you will implement all the other screen sizes the exact same way you'll be implementing the medium device tag.

Let us explain each of these concepts to help you better understand why we write these:

1. Col

- The “col” keyword stands for column. It’s how Bootstrap identifies that you are going to want to work with columns.

2. sm/md/lg/xl

- This is the screen size that we are going to be working with. As mentioned, you will typically be working with **md** through the bulk of the course; however, it’s encouraged that you play around with other values to see how they will change the display of your page.

3. *

- The asterisk is actually a wild-card placeholder! This will eventually be replaced with a number that will describe how many columns it will take up (i.e. if we enter the number six, it will take up half of the page – **remember**: you only have 12 columns to work with!).

ROWS

Now that you have a better understanding of different screen sizes and columns, we need to discuss one last concept that is an important part of Bootstrap: the implementation of rows.

12											
6						6					
3			3			3			3		
1	1	1	1	1	1	1	1	1	1	1	1

In the grid pyramid image that shows all the columns in different colours, you will notice that each number is associated with a different “line”. These lines are known as rows.

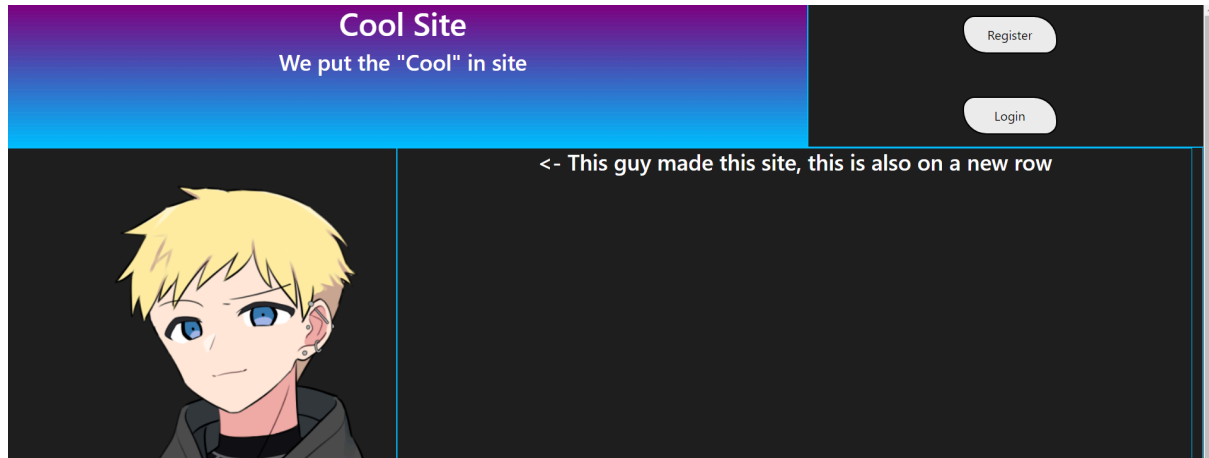
Unlike columns, there can be an infinite number of rows (as a web page can reach an unlimited height).

PUTTING IT ALL TOGETHER

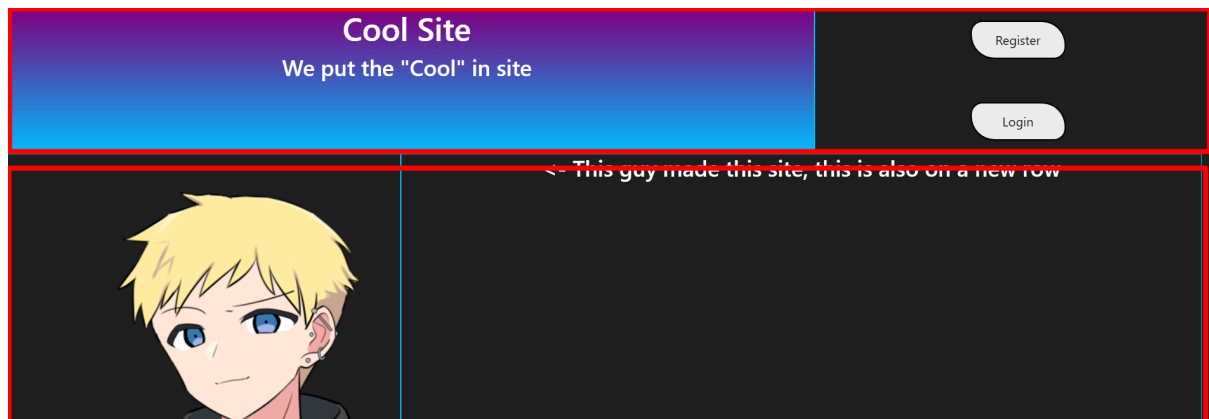
Now that we know that websites are built using the grid system, we can start making use of Bootstrap. Something that makes Bootstrap so powerful for new learners is that they only have to learn class names. It adds no additional HTML tags or new systems that make it a complex learning experience.

It's suggested that you create a text document that contains a cheat sheet of all the class names so you can avoid having to come back to this PDF file every time you need to remember the spelling of a term.

Let's start by creating a basic web page using Bootstrap. Below is a screenshot of the final product (it's not the most beautiful design, but it does its job providing an example to showcase the concept of Bootstrap.)



Let's start by discussing where the rows are in this project.



Notice how each major part of the website is on its own row. So let's create the code that will be used to make the rows!

```
<body>
  <div class="row">
  </div>
  <div class="row">
  </div>
</body>
```

You will notice that we have created two divs (a div for each row on the website). You'll notice that we've given each div tag a "class" name with the term "row". It's important to note that this is a class name created by Bootstrap, which will start to modify how your website will be displayed. This is how it separates each row on the website.

Now that we've created our two rows, let's look at the image below to see how many columns your website will be made up of:



You will notice that inside each section, there are either eight or four different columns. We decide how many we want. So, let's head over to our code and put our columns in place.

```
<div class="row">
  <div class="col-md-8">
    <!-- this is col 1-8 (this is a cool site)-->
  </div>
  <div class="col-md-4">
    <!-- This is col 9-12 (login/register button) -->
  </div>
</div>
<div class="row">
  <div class="col-md-4">
    <!-- This is col 1-4 (The image) -->
  </div>
  <div class="col-md-8">
    <!-- This is col 5-12 (The text) -->
  </div>
</div>
```

As you can see, we've now added a div inside our "row" divs and given them a class name of "col-md-*". You can see how the different screen sizes will come into play here as you can change the "md" to a different size based on the device you expect the user to use.

From there, you can enter the total number of columns you want each div to take. That's the number after the device size. What's important to remember when using these is that the total should always be less than or equal to 12. So make sure you check your calculations when you start modifying the sizes.

Once everything is written, you can now start adding any normal code (such as images and buttons) inside these div tags!

```
<div class="row">
  <div class="col-md-8 background">
    <h1>Cool Site</h1>
    <h3>We put the "Cool" in site</h3>
  </div>
  <div class="col-md-4">
    <button>Register</button>
    <br>
    <br>
    <button>Login</button>
  </div>
</div>
<div class="row">
  <div class="col-md-4">
    
  </div>
  <div class="col-md-8">
    <h3><- This guy made this site, this is also on a new row</h3>
  </div>
</div>
```

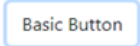
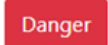
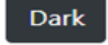
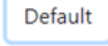



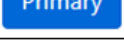
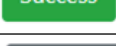


Once you've finished up all your code, you have officially created your first ever responsive website! This is a huge step in your programming journey and you should be proud of what you have accomplished.

BOOTSTRAP BUTTONS AND STYLING

Bootstrap has a lot of features built into it, most of which we can control with the class name. One of these features is the different styling of buttons. Below are a couple of examples.

For more examples, feel free to visit:

- <https://getbootstrap.com/docs/5.3/getting-started/introduction/#quick-start>
- <https://bootstrapshuffle.com/classes>

Class	Description	Example
<code>.btn</code>	Adds basic styling to any button	
<code>.btn-danger</code>	Indicates a dangerous or potentially negative action	
<code>.btn-dark</code>	Dark grey button	
<code>.btn-default</code>	Indicates a default/standard button	
<code>.btn-info</code>	Contextual button for informational alert messages	
<code>.btn-light</code>	Light grey button	
<code>.btn-link</code>	Makes a button look like a link (will still have button behavior)	
<code>.btn-primary</code>	Provides extra visual weight and identifies the primary action in a set of buttons	
<code>.btn-success</code>	Indicates a successful or positive action	
<code>.btn-secondary</code>	Indicates a "less" important action	
<code>.btn-warning</code>	Indicates caution should be taken with this action	

Bootstrap is a powerful tool that can be used to create amazing websites with a lot less effort than manually creating a responsive website.

Instructions

Read and run the accompanying example files provided before doing the task, to become more comfortable with the concepts covered in this task.



Take note:

The task(s) below is/are **auto-graded**. An auto-graded task still counts towards your progression and graduation.

Give it your best attempt and submit it when you are ready.

You will receive a 100% pass grade once you've submitted the task.

When you submit the task, you will receive an email with a link to a model answer, as well as an overview of the approach taken to reach this answer. Take some time to review and compare your work against the model answer. This exercise will help solidify your understanding and provide an opportunity for reflection on how to apply these concepts in future projects.

In the same email, you will also receive a link to a survey for this task, which you can use as a self-assessment tool. Please take a moment to complete the survey.

Once you've done that, feel free to progress to the next task.

Auto-graded Task

Follow these steps:

- Use CSS and the Bootstrap framework to style and position the elements on your webpage that you created in the HTML task (**index.html**) as attractively as possible.
- Ensure that you apply the following styling to your CV:
 - **Colour:** change the background colour of the entire page.
 - **Font:** modify the font according to your preference. You're welcome to change the font family, colour, size, weight, and style.
 - **Image size:** adjust the size of the image. Feel free to modify the width, height, and border-radius properties as you please.
 - **Positioning:** centralise the navbar and place the elements horizontally next to each other. Ensure to adjust and space out the

content appropriately in a presentable manner. You may consider using the margin and padding property.

- **Layout:** use a flexbox OR grid layout to make your CV responsive to different screen sizes. You could use Bootstrap's grid system to structure sections of your CV into rows and columns OR you could use flexbox classes for more flexible alignment of items along a single axis.
- Ensure all properties are visually pleasing in appearance, as this plays a role in positively influencing user experience.



Rate us

Share your thoughts

Hyperion 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.

