# Bootstrap

[﻿Bootstrap](https://getbootstrap.com/) is a free CSS library that is very common across the internet. It is nice for many reasons, but one thing that it does well is allow you to quickly create well-formatted content and useful snippets that can be used in your learning modules. If Bootstrap is installed by your LMS, then you will have access to many of the formatting options and rich features provided by Bootstrap below (and more).

## Basic (CSS-Only) Elements

### [Alerts](https://getbootstrap.com/docs/5.3/components/alerts/)

There are no built-in alert elements for HTML, but sites often need some method for designating alerts. Bootstrap does this by assigning p, div, or other elements with the alert class followed by a color formatting class, such as alert-success or alert-danger.

class="alert alert-primary"

class="alert alert-secondary"

class="alert alert-success"

class="alert alert-danger"

class="alert alert-warning"

class="alert alert-info"

class="alert alert-light"

class="alert alert-dark"

### [Backgrounds](https://getbootstrap.com/docs/5.3/components/buttons/)

Backgrounds of elements can be set by changing the class of the elements to one of the following values:

.bg-primary

.bg-primary-subtle

.bg-secondary

.bg-secondary-subtle

.bg-success

.bg-success-subtle

.bg-danger

.bg-danger-subtle

.bg-warning

.bg-warning-subtle

.bg-info

.bg-info-subtle

.bg-light

.bg-light-subtle

.bg-dark

.bg-dark-subtle

.bg-body-secondary

.bg-body-tertiary

.bg-body

.bg-black

.bg-white

.bg-transparent

### [Buttons](https://getbootstrap.com/docs/5.3/components/buttons/)

This is a normal HTML button:

Click here

And these are examples of Bootstrap buttons:

btn btn-primary btn btn-secondary btn btn-success btn btn-danger btn btn-warning btn btn-info btn btn-light btn btn-dark btn btn-link

Basic formatting for Bootstrap buttons (e.g., size, spacing) is achieved by adding the class btn to the button HTML element. Then, you can add a second class to adjust the color, such as btn-primary or btn-secondary.

### Font Colors

Font colors of elements can be set by changing the class of the elements to one of the following values (note that this can also be combined with background classes to give light text a dark background):

.text-primary

.text-primary-emphasis

.text-secondary

.text-secondary-emphasis

.text-success

.text-success-emphasis

.text-danger

.text-danger-emphasis

.text-warning

.text-warning-emphasis

.text-info

.text-info-emphasis

.text-light

.text-light-emphasis

.text-dark

.text-dark-emphasis

.text-body

.text-body-emphasis

.text-body-secondary

.text-body-tertiary

.text-black

.text-white

.text-black-50

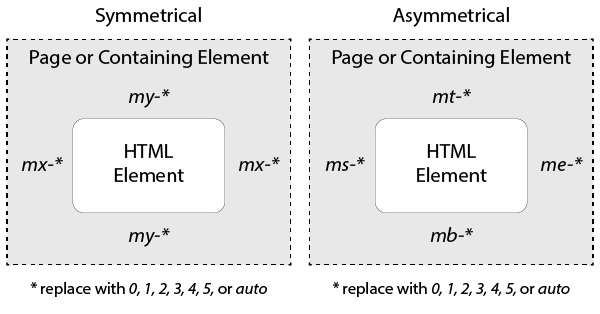
.text-white-50

### [Margins](https://getbootstrap.com/docs/5.3/utilities/spacing/#margin-and-padding)

Margins are the white space around an element. These values can be set either symmetrically (by referencing the x- and y-axis margins) or asymmetrically (by referencing the top, bottom, start, or end margin) by applying classes to an element as follows:

Figure 1

Bootstrap Margin Classes



Symmetrical references are just shorter forms of writing asymmetrical references. So, a class of mx-2 would be equivalent to a class value of ms-2 me-2. The auto definition attempts to automatically layout the page, such as by centering the content. You can also apply margins to all four sides of an element with the m-\* class.

Here are a few examples:

m-1

m-2

m-3

mx-1 my-4

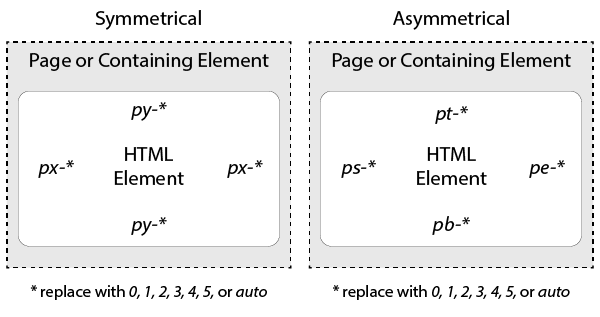
mt-1 me-2 mb-3 ms-4

### [Padding](https://getbootstrap.com/docs/5.3/utilities/spacing/#margin-and-padding)

Padding is the white space within an element separating an element's content from its own border. These values can be set either symmetrically (by referencing the x- and y-axis padding) or asymmetrically (by referencing the top, bottom, start, or end padding) by applying classes to an element as follows:

Figure 1

Bootstrap Margin Classes



Symmetrical references are just shorter forms of writing asymmetrical references. So, a class of px-3 would be equivalent to a class value of ps-2 pe-2. The auto definition attempts to automatically layout the page, such as by centering the content. You can also apply padding to all four sides of an element with the p-\* class.

Here are a few examples:

p-1

p-2

p-3

px-1 py-4

pt-1 pe-2 pb-3 ps-4

### Pro Tip on Padding

When laying out text and other elements, it typically looks better for the x padding to be a bit larger than the y padding. So, if you have a paragraph with a py-1, then you'll likely want a px-2 to go with it.

### Progress Bars

Progress bars are normally made in HTML by simply placing one div element inside of another, with the outer element serving as the empty background and the inner element serving as the progress indicator. By changing the width value of the inner div element's style attribute to a percent of the whole, you can change the visible progress. You can also change the color of either the inner or outer element by changing its class to a background class.

### Shadows

Shadows are useful to convey depth to elements on a page, though they should be used intentionally. The shadow classes include shadow-sm, shadow, and shadow-lg and have the following effect when added to an element:

.shadow-sm

.shadow

.shadow-lg

### [Text Alignment](https://getbootstrap.com/docs/5.3/utilities/text/#font-size)

Content can be horizontally aligned using the following classes: text-start, text-center, or text-end.

.text-start

.text-center

.text-end

Note: Previous versions of Bootstrap used .text-left and .text-right, but since not all languages go from left-to-right, shifting to start and end allows formatting to be better applied across all languages.

### Text Size

When resizing text, you should first make sure that you are resizing semantically. So, if you want to resize the text because it is a heading, make it a heading (e.g., H1, H2) rather than just changing the size.

There are cases where you may still want to resize text that is not in a heading. In these cases, you can use the following font-size classes, which correspond to the heading sizes:

.fs-1

.fs-2

.fs-3

.fs-4

.fs-5

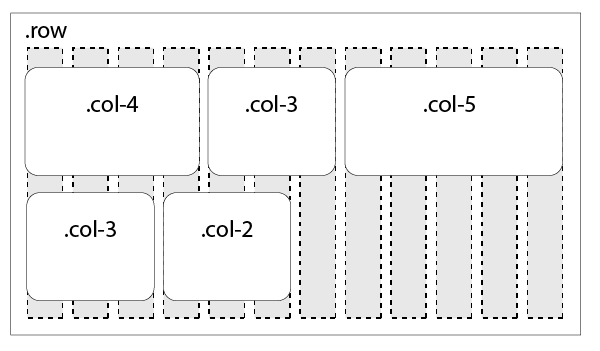
.fs-6

## Horizontal Layout Elements

Laying out content on a web page horizontally can be problematic, because screen sizes vary and accessibility and machine tools (like screenreaders) may have difficulty understanding the order of content. In the early days of the internet, people often relied on tables to layout content into grids, but this leads to a variety of problems. Instead, current web design tends to use flexible columns and floats to achieve complex horizontal layouts.

### Grid Columns

By creating an element in Bootstrap and assigning it the class row, Bootstrap will organize all content in that element into 12 possible columns. Then, you can assign the class col-1, col-2, ... col-12 to any elements within the row element to have it layout the element as one of the columns.



Here is an example of a simple 2-column layout:

<div class="row">  
  <p class="col-6">Here's the first column.</p>  
  <p class="col-6">Here's the second column.</p>  
</div>

Here's the first column.

Here's the second column.

If we wanted three columns, we could divide the 12 column options however we like between them, such as the following:

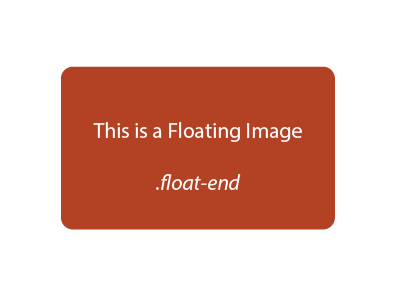
<div class="row">  
  <p class="col-4">Here's the first column.</p>  
  <p class="col-4">Here's the second column.</p>  
  <p class="col-4">Here's the third column.</p>  
</div>

Here's the first column.

Here's the second column.

Here's the third column.

### Floats

Floats can be used to quickly align an element to the start or end of its parent element and have other elements wrap around it. Floats are most commonly used with images, such as the image floating to the right of this text area.

To apply a float to an element, just add the float-start or float-end class to it.

Note that elements will continue to wrap around the float until they are told to stop. You can do this by giving the next element the clearfix class or by making a new empty element of any kind, such as a div, and giving it the class clearfix.

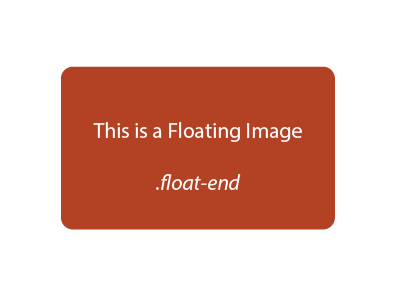
### Widths

Bootstrap provides a shortcut for assigning common widths to elements with the w-25, w-50, w-75, and w-100 classes, which give the element the value width as a percent of the width of the parent element. So, w-75 would be equivalent to style="width: 75%;". This can be particularly helpful in conjunction with floats by allowing you to define an image as 50% width via w-50 and floating it at the end of a paragraph via float-end.

### Flexible Layouts

And since not all screens will be the same size, we need horizontal layouts to react to the width of the screen. Though we might want an image to float to the right on a large screen, we might want the text wrapping to go away on a small screen (so that we have more than two words of text per line.

To achieve this, we can include references to the user's current screen size to any of the layout elements above to apply it to only certain screen sizes, such as sm, md, lg, xl, and xxl. These are called breakpoints. Because modern web practices have a mobile-first mindset, classes will apply in a manner that assume you are applying formatting at the smallest possible screen size unless you explicitly apply it only to a larger screen size. So, md will also apply to screen widths of lg, xl, and xxl, but xxl will not apply to any other screen widths.



So, if you apply a class of float-xl-end, then the element will float only on screen widths of xl or larger. You can test this in your browser by resizing this page and noticing that the floating image to the right no longer floats once the screen gets smaller. Contrast this with the floating image above that always floats no matter the screen size.

These width modifiers can be applied to the other horizontal layout elements above to achieve very powerful layouts. For instance, you could apply col-12 col-md-6 to an element to make it six columns wide on md and larger screens, and 12 columns wide on smaller screens. Or, you could make an image 50% width on larger screens and 100% width on smaller screens by assigning it the classes w-100 w-lg-50.

### Display

You can also decide to completely hide or show content based on the user's screen size by using the d-\* (or display) classes. Some of the more common uses of this are to alternate between d-block (to have an element fill the screen horizontally) and d-none (to hide the element). For instance, by default a p element is using a block display, so applying d-block to it won't change anything, but if you give a p element a d-none class, then it will be hidden.

You can combine these classes with the breakpoints above to have elements show or hide themselves depending upon the size of the screen. So, an alert message with classes d-none d-md-block will only reveal itself once the screen reaches the md breakpoint size. Or, an image with classes d-lg-none will hide itself once the screen reaches the lg breakpoint.

You can also have elements only show themselves under certain conditions, like if the page is being printed. For instance, an image with the class d-print-none won't be visible when you print the page, while a paragraph with the class d-none d-print-block will only be shown when the page is being printed.

## Advanced (CSS + JS) Elements

### Accordion

Accordion Item #1

**This is the first item's accordion body.**

It is shown by default, until the collapse plugin adds the appropriate classes that we use to style each element. These classes control the overall appearance, as well as the showing and hiding via CSS transitions. You can modify any of this with custom CSS or overriding our default variables. It's also worth noting that just about any HTML can go within the

.accordion-body

, though the transition does limit overflow.

Accordion Item #2

**This is the second item's accordion body.**

It is hidden by default, until the collapse plugin adds the appropriate classes that we use to style each element. These classes control the overall appearance, as well as the showing and hiding via CSS transitions. You can modify any of this with custom CSS or overriding our default variables. It's also worth noting that just about any HTML can go within the

.accordion-body

, though the transition does limit overflow.

Accordion Item #3

**This is the third item's accordion body.**

It is hidden by default, until the collapse plugin adds the appropriate classes that we use to style each element. These classes control the overall appearance, as well as the showing and hiding via CSS transitions. You can modify any of this with custom CSS or overriding our default variables. It's also worth noting that just about any HTML can go within the

.accordion-body

, though the transition does limit overflow.

Read this online at <https://edtechbooks.org/elearning_hacker/bootstrap>