Bootstrap 4

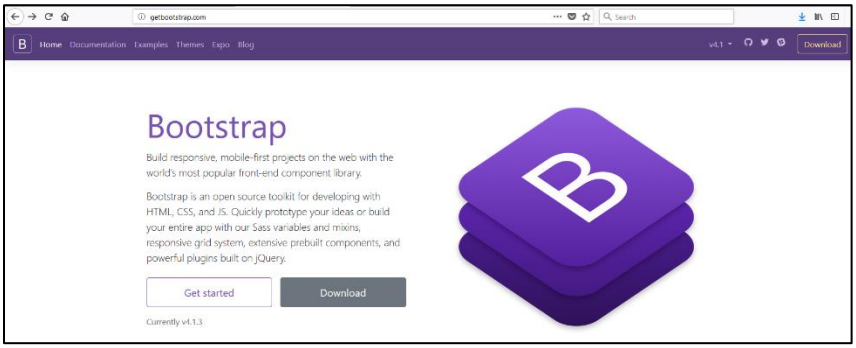
Using Basic Styles in Bootstrap 4

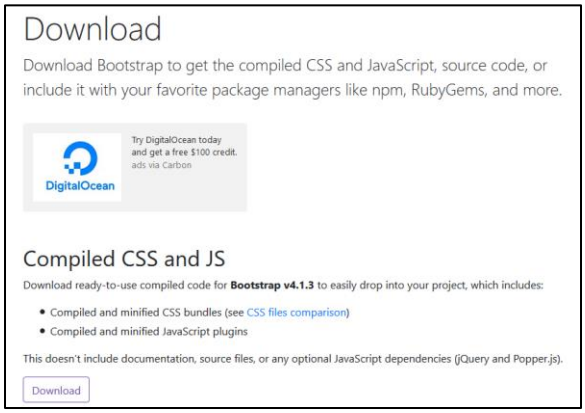
Installing Bootstrap 4

Bootstrap is a free and open-source front-end framework (library) for designing websites and web applications. It contains HTML- and CSS-based design templates for typography, forms, buttons, navigation and other interface components, as well as optional JavaScript extensions.

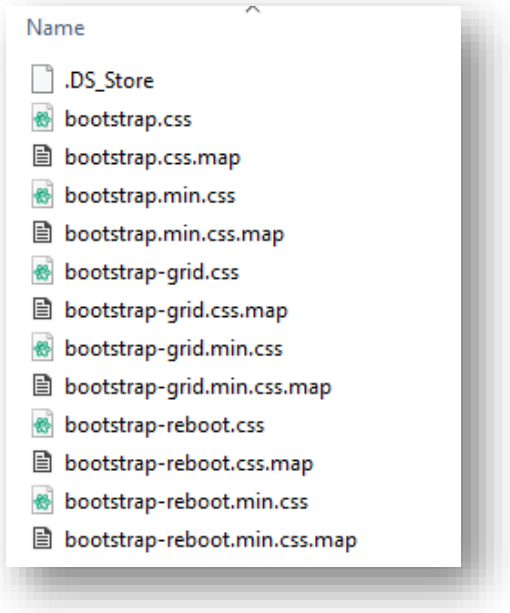
Installation

This version of Bootstrap has a lot of different installation options. So let's take a quick look at what we have available. Now you can install Bootstrap in one of four ways. The first way is to simply download the pre compiled Bootstrap CSS and JavaScript files. It works really well if you need to install a copy of Bootstrap that will work even without an internet connection. Another way is to use CDNs. A CDN is a content delivery network, which needs a place that hosts common libraries like Bootstrap. You can also download the source files. This way, you download not just the CSS and JavaScript, but all the files that the developers use to create Bootstrap.
Now to get instructions and to download a copy of Bootstrap manually, first go to the getbootstrap.com website right here and then click on this Download button.



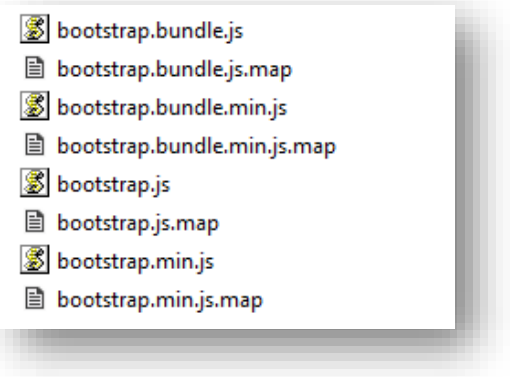


Now this is not going to download anything. It's going to take you to another page where it shows all the different download options. If you scroll down you can see the first option is Compiled CSS and JavaScript. Let's go ahead and click on this button because we want decompiled and minified CSS versions of the library.



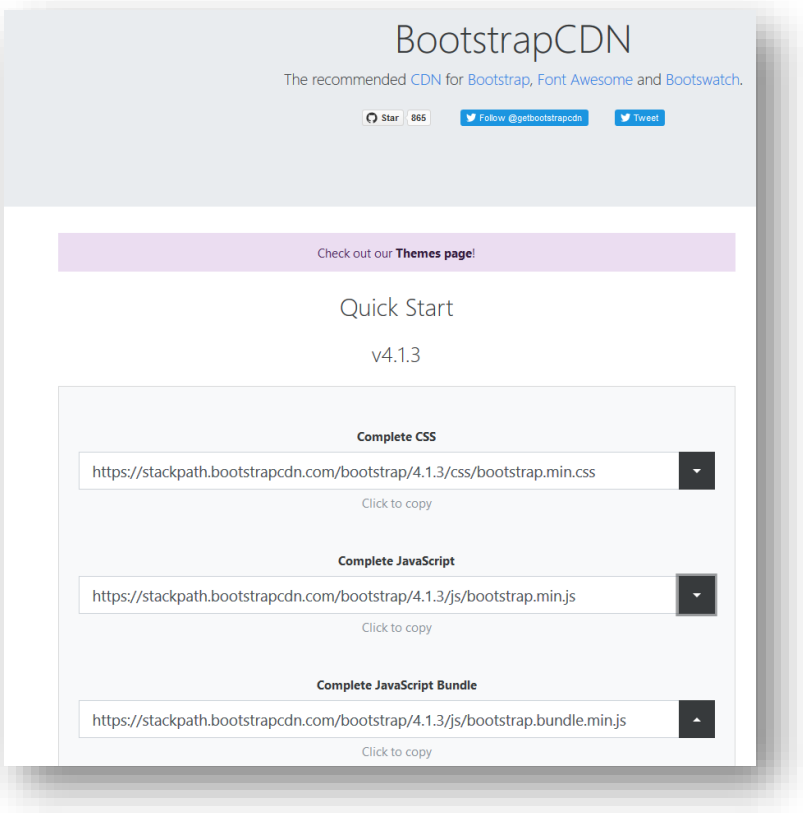
If you open up the folder, you see a CSS folder plus a JavaScript folder. The CSS folder has a number of different versions of the CSS use for the Bootstrap framework. Now if you look at this folder, you'll notice that there are three types of files: regular css files, minimized version, and map files. Map files are files that are useful if you're using Bootstrap in development mode so that they could point you not to the CSS, but to the original SAS code that was used to create the CSS. Some people like to use only the grid features of Bootstrap for a layout so you can get just that if you want to. There's also a Bootstrap reboot file. The reboot files are the special Bootstrap code that resets the CSS and browsers so that they work more consistently in different platforms.

The regular version of Bootstrap contains the grid, the reset, and everything else in Bootstrap. You can choose the regular version if you're going to customize Bootstrap, or just choose Bootstrap.min.css if you just want to use that file.



In the JavaScript folder, you'll also find several files as well. JavaScript also has maps. Since most people write in a version of JavaScript, it gets translated so that it works with older browsers. There are regular and minimized versions of the JavaScript as well. Other than that, there are two types of files: the regular Bootstrap and the Bootstrap bundle. The Bootstrap bundle has all the Bootstrap code plus an additional library called 'popper.js'. **Popper.js** is a positioning engine, its purpose is to calculate the position of an element to make it possible to position it near a given reference element.

Most of time you're going to need just the regular Bootstrap file, however you also use the minimized version since you don't want to edit any of the Bootstrap JavaScript yourself.



When someone visits a site that uses a CDN link, their browser will check its cache or memory to see if the visitor has been to a similar site that's also using the same link. If that's the case, then the browser will load the cached version of the library. Since it's already stored in memory, that makes the new site load faster since the browser will not have to request the file. You can download the CDN version of bootstrap at https://www.bootstrapcdn.com/

All together

In order to get start a project using bootstrap, you will need the following file links to your project document:

- Bootstrap
- jQuery
- popper.js
- basic HTML document

To get all the templates at once, you can go to: http://getbootstrap.com/docs/4.0/getting-started/introduction/ and get a preset code.

Starter template

Be sure to have your pages set up with the latest design and development standards. That means using an HTML5 doctype and including a viewport meta tag for proper responsive behaviors. Put it all together and your pages should look like this:

Copy to clipboard

Copy

```
<!doctype html>
<html lang="en">
  <head>
    <!-- Required meta tags -->
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

    <!-- Bootstrap CSS -->
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css" integrity="sha384-Gn5

    <title>Hello, world!</title>
  </head>
  <body>
    <h1>Hello, world!</h1>

    <!-- Optional JavaScript -->
    <!-- jQuery first, then Popper.js, then Bootstrap JS -->
    <script src="https://code.jquery.com/jquery-3.2.1.slim.min.js" integrity="sha384-KJ3o2DKtIkvYIK3UENzmM7KCkRr/rE9/Qpg6aAZG
    <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.12.9/umd/popper.min.js" integrity="sha384-ApNbgh9B+Y1QKtv
    <script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstrap.min.js" integrity="sha384-JZR6Spejh4U02d8jOt6vL
  </body>
</html>
```

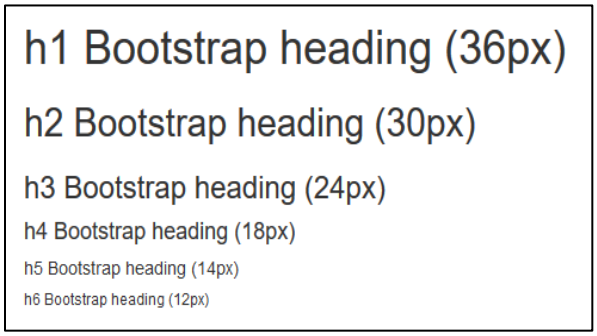
Text/Typography

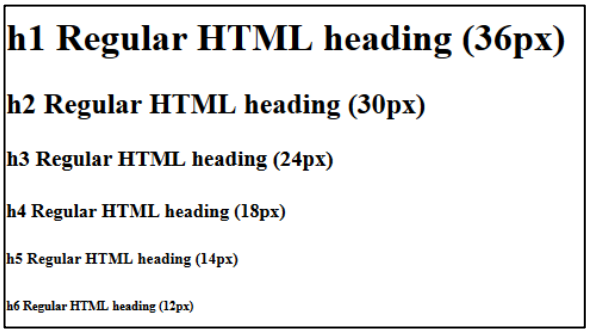
The CSS code that comes with Bootstrap overrides default browser behavior. Besides making things look great, it's also designed to be easy to override with your own CSS.

Bootstrap 4 uses a default **`font-size`** of 16px, and its **`line-height`** is **1.5**.

The default **`font-family`** is "Helvetica Neue", Helvetica, Arial, sans-serif.

In addition, all `<p>` elements have **`margin-top: 0`** and **`margin-bottom: 1rem`** (16px by default).





You can try some to the HTML code in bootstrap:

- **`<small>`** ➔ element is used to create a lighter, secondary text in any heading.

  ```
  <h1>h1 heading <small>secondary text</small></h1>
  ```

  h1 heading secondary text

- **`<mark>`** ➔ element in used to highlight text

  ```
  <p>Use the mark element to <mark>highlight</mark> text.</p>
  ```

  Use the mark element to highlight text.

- **`<kbd>`** ➔ element indicates input that is typically entered via the keyboard

  ```
  <p>Use <kbd>ctrl + p</kbd> to open the Print dialog box.</p>
  ```

  Use `ctrl + p` to open the Print dialog box.

Bootstrap also has some contextual classes.

- **`display-1, display-2, display-3, display-4`** ➔ class is an almost twice the regular size
- **`text-nowrap`** ➔ class lets the text of a paragraph or any other element sort of fit to the left and the right with the same amount of spacing.
- **`Text`** ➔ class allows to set the position itself or size and position

There is also other classes related to text: **`text-lowercase, text-uppercase, text-capitalize, font-weight-bold, font-weight-normal, font-italic, text-justify`**

You can check more bootstrap typography at [w3schools]

Lists and blockquotes

Bootstrap provides some classes that help you work with common CSS tags like lists and blockquote elements.

- **`list-unstyled`** ➔ displays the list without bullets
- **`list-inline-item`** ➔ display the element in line

The **`<blockquote>`** tag specifies a section that is quoted from another source.

Browsers usually indent **`<blockquote>`** elements. In bootstrap, the class blockquote makes the font a little bigger and it also gives a vertical line within the quoted text.

You can also create a `blockquote-footer` it creates a nice format around the wrapped text.

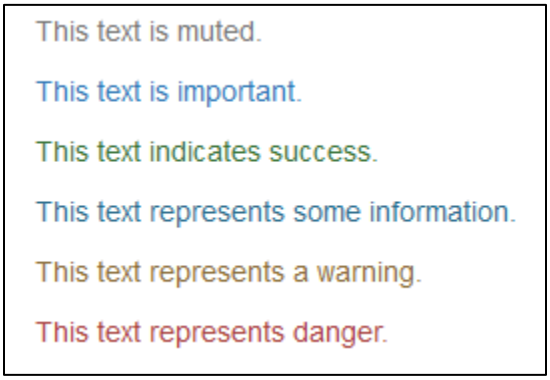
`blockquote class=”blockquote-reverse”` aligns the quoted section to the right.

Colors and Image Manipulation using Bootstrap 4

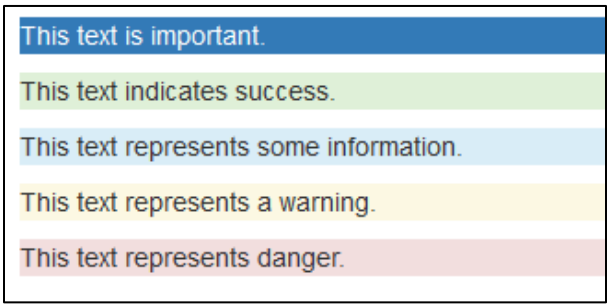
Colors

Bootstrap has a number of colors that you can access through contextual color names. They're used everywhere, including in buttons, background colors, as well as texts, so you'll hear some of these names in different places, but let's take a look at the basic options available in texts, links, and backgrounds. So the way that these work is by using the prefix text, and then using one of the contextual color names.

The classes for text colors are **:.text-muted, .text-primary, .text-success, .text-info, .text-warning, and .text-danger**



The classes for background colors are: **bg-primary, bg-success, bg-info, bg-warning, bg-danger**



Working with images

Bootstrap also has some very useful classes that help you work with images, as well as with figures.

Border images

The `rounded` class adds rounded corners to an image. You can also apply direction of which edge the image do you want it to be rounded.

`rounded`, `rounded-DIR` round edges

`top` `right` `bottom` `left`

`circle` `rounded-0`

The **`rounded-circle`** ➔ class shapes the image to a circle

The **`img-thumbnail`** ➔ class shapes the image to a thumbnail (bordered)

Aligning images

Float an image to the right with the **`float-right`** class or to the left with **`float-left`**

Center an image by adding the utility classes **`mx-auto`** (margin:auto) and **`d-block`** (display:block) to the image.

Responsive images

Responsive images automatically adjust to fit the size of the screen.

- **`img-fluid`:** Create responsive images by adding an **`img-fluid`** class to the **`<img>`** tag. The image will then scale nicely to the parent element.

  The **`img-fluid`** class applies **`max-width: 100%;`** and **`height: auto;`**

- **`figure-img`** class makes the text with a smaller font and lighter color. The **`<figure>`** tag in HTML specifies self-contained content, like illustrations, diagrams, photos, code listings, etc.

  While the content of the **`<figure>`** element is related to the main flow, its position is independent of the main flow, and if removed it should not affect the flow of the document.

- **`figure-caption`** class makes the text in **`<figcaption>`**with a smaller font and lighter color. The HTML **`<figcaption>`** element represents a caption or legend for the rest of the contents its parent figure element, if any.

CSS variables

Bootstrap has some features that allows you to work with a new feature of CSS called CSS Variables. Now CCS Variables are a newer feature within CSS that allows you to store a set of property values that you can reuse within your CSS. Now this version of Bootstrap provides pre-written CSS variables that can be used in developing your projects and customizing your colors. CSS Variables are so new that browser support is an issue. They're not supported in any version of Internet Explorer, and only on very new versions of the other browsers.

Syntax

```
var(custom-name, value)
<style>
  :root{
    --hello:#25FF64;
    }
  </style>
  <h1 style="color:var(--hello);">Our Commitment</h1>
```

Mastering Layout with bootstrap

By far the most useful part of Bootstrap is the grid, so this is the first thing you should master. The Bootstrap grid gives you a framework to create any layout that you can think of. Now, Bootstrap layouts have three main pieces. The first is containers, which can be either responsive fixed width that snap to certain breakpoints, or completely fluid containers which take up 100% of the width of the view port, which is the browser window or the width of your device. Inside containers, you can also place content in rows and columns.

Bootstrap uses a 12-column grid system that has breakpoints for extra small, small, medium, large, and extra large devices. If you're coming from a previous version of Bootstrap, note that there is an additional breakpoint for smaller devices. The grid is extremely flexible. You can create multiple layouts for different breakpoints, reset, offset, nest, and even customize the order of the columns at different breakpoints. Bootstrap's grid is very powerful.

If you can think of a layout, you can build it easily and responsively with the Bootstrap grid. The more time you spend learning how to use it, the easier it will be to get your work done.

Containers and rows

One of the main reasons people use Bootstrap is to have access to an amazing grid that makes layout easy to do and responsive. This is one of the most important things you should master in Bootstrap, so let's take a look. First of all, the grid is a responsive 12-column system for creating just about any layout you can think of. It uses a technology called Flexbox that makes it easier to create complex layout with minimal code. In order to work with the grid, you need to master three simple components.

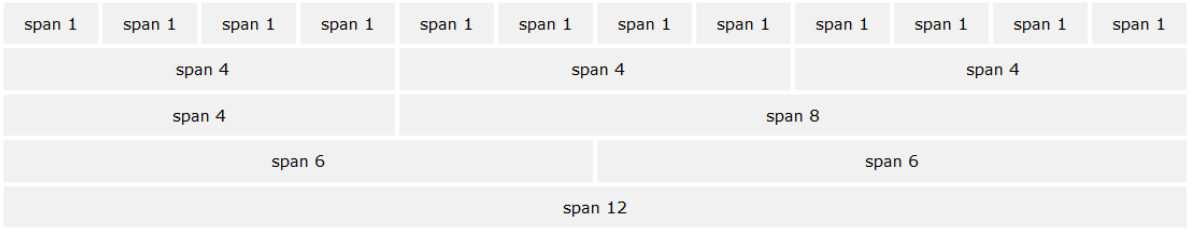
The first is containers, which can be used with or without the grid to **`align-content`** either to the viewport, or center it around a set of breakpoints. Next is rows and columns. They work together to allow you to create the layouts. The rows prepare the columns for layout, and the columns are complex and extremely flexible, so we'll cover them in a separate video in detail. So first, let's go ahead and take a look at containers. There are two different types of containers.

`container` class is used for regular containers, which center content and snap to certain grid points.

`container-fluid` are fluid containers, which are always the full width of the viewport, which means the width of the device or the browser window. One of the reasons you use a container is because you get a 15 pixel padding on each side to make sure it works well with backgrounds and other elements. Now here's the breakpoints that the regular container will adjust to. First is anything smaller than 576 pixels, and so the bootstrap grid adjusts to content that is smaller than 576 pixels, and then some of these other breakpoints.

Rows and columns

Bootstrap's grid system allows up to 12 columns across the page. If you do not want to use all 12 column individually, you can group the columns together to create wider columns:



Bootstrap's grid system is responsive, and the columns will re-arrange depending on the screen size: On a big screen it might look better with the content organized in three columns, but on a small screen it would be better if the content items were stacked on top of each other.

Grid Classes

The Bootstrap 4 grid system has five classes:

- **`.col`**– (extra small devices - screen width less than 576px)
- **`.col-sm`**– (small devices - screen width equal to or greater than 576px)
- **`.col-md`**– (medium devices - screen width equal to or greater than 768px)
- **`.col-lg`**– (large devices - screen width equal to or greater than 992px)
- **`.col-xl`**– (xlarge devices - screen width equal to or greater than 1200px)

The classes above can be combined to create more dynamic and flexible layouts.

Tip: Each class scales up, so if you wish to set the same widths for **`sm`** and **`md`**, you only need to specify **`sm`**.

Screen Breakpoints

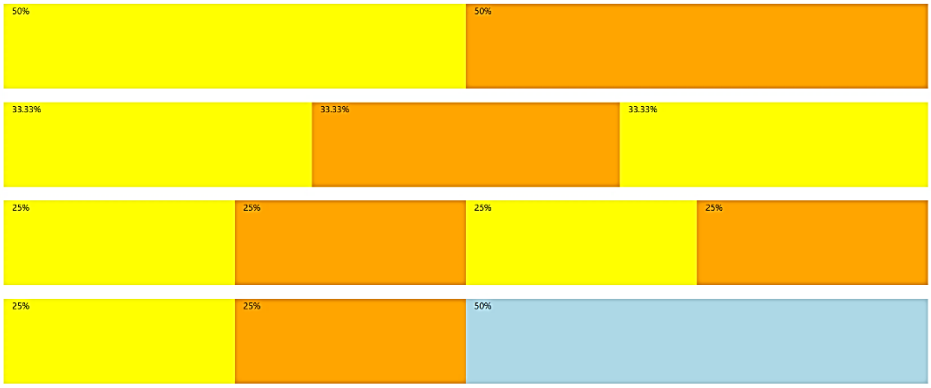
Grid System Rules

Some Bootstrap 4 grid system rules:

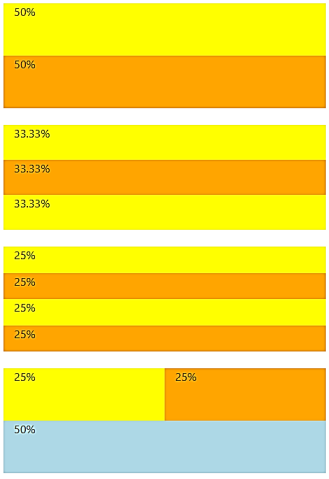
- Rows must be placed within a **`.container`** (fixed-width) or **`.container-fluid`** (full-width) for proper alignment and padding
- Use rows to create horizontal groups of columns
- Content should be placed within columns, and only columns may be immediate children of rows
- Predefined classes like **`.row`** and **`.col-sm-4`** are available for quickly making grid layouts
- Columns create gutters (gaps between column content) via padding. That padding is offset in rows for the first and last column via negative margin on .rows
- Grid columns are created by specifying the number of 12 available columns you wish to span. For example, three equal columns would use three **`.col-sm-4`**
- Column widths are in percentage, so they are always fluid and sized relative to their parent element

Example) Create row with different width:

Computer view



Smartphone view



HTML

```
<div class="container-fluid">
    <!-- Control the column width, and how they should appear on different
devices -->
    <div class="row">
      <div class="col-sm-6" style="background-color:yellow;">50%</div>
      <div class="col-sm-6" style="background-color:orange;">50%</div>
    </div>
    <br>

    <div class="row">
      <div class="col-sm-4" style="background-color:yellow;">33.33%</div>
      <div class="col-sm-4" style="background-color:orange;">33.33%</div>
      <div class="col-sm-4" style="background-color:yellow;">33.33%</div>
    </div>
    <br>

    <!-- Or let Bootstrap automatically handle the layout -->
    <div class="row">
      <div class="col-sm" style="background-color:yellow;">25%</div>
      <div class="col-sm" style="background-color:orange;">25%</div>
      <div class="col-sm" style="background-color:yellow;">25%</div>
      <div class="col-sm" style="background-color:orange;">25%</div>
    </div>
    <br>

    <div class="row">
      <div class="col" style="background-color:yellow;">25%</div>
      <div class="col" style="background-color:orange;">25%</div>
      <div class="col-sm-6" style="background-color:blue;">50%</div>
    </div>
```

To take full advantage of the 12 column grid, you have to master the use of columns.

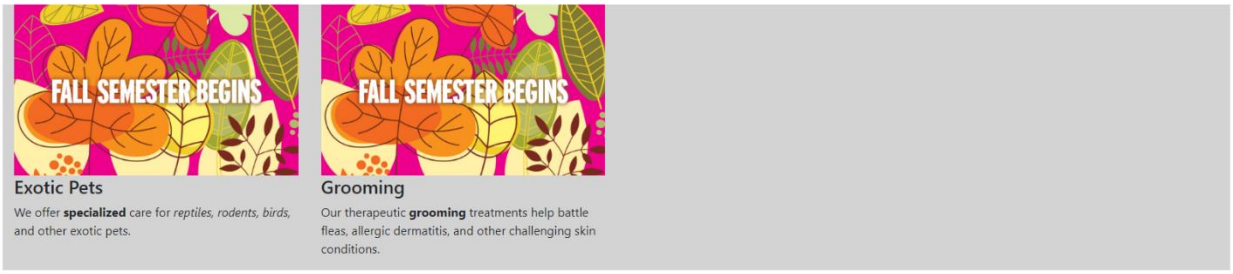
Multiple columns

The grid in Bootstrap is powerful because of the many ways that we can use it. One thing we can do is specify multiple-column breakpoints for each of our grid elements.

	Extra small < 576px	Small ≥ 576px	Medium ≥768px	Large ≥992px	Extra large ≥1200px
Prefix	.col-	.col-sm-	.col-md-	.col-lg-	.col-xl-

Offsetting columns

Another way to control the grid is to offset columns in order to move their position on the grid. For example, a grid without offset looks as the following:



HTML

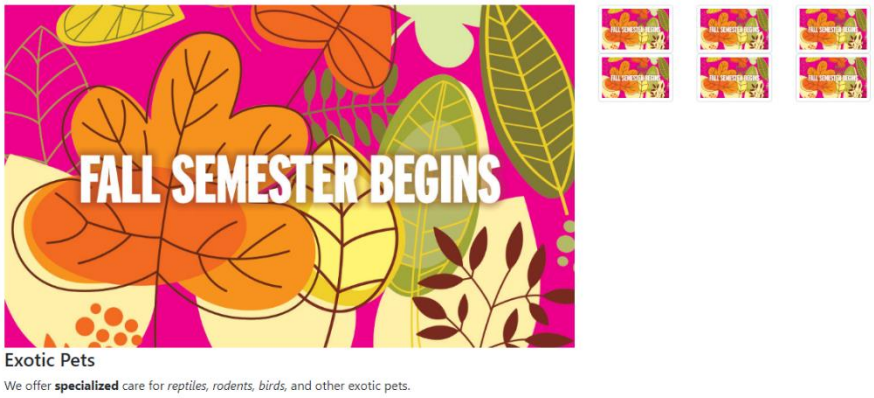
```
<div class="row" style="background: lightgray;">
   <section class="col-md-3" >
       <img src="images/image.jpg" alt="sample image" style="width:100%;" >
       <h4>Exotic Pets</h4>
       <p>We offer <strong>specialized</strong> care for <em>reptiles,
      rodents, birds,</em> and other exotic pets.</p>
   </section>
   <section class="col-md-3">
       <img src="images/image.jpg" alt="sample image" style="width:100%;" >
       <h4>Grooming</h4>
       <p>Our therapeutic <span class="font-weight-bold">grooming</span>
      treatments help battle fleas, allergic dermatitis, and other challenging
      skin conditions.</p>
   </section>
</div><!-- end of row -->
```

`offset` class in columns is used to shift but the number of columns over. From the previous example, if we add offset to **`<section class="col-md-3 offset-sm-1">`** the webpage will look as the following:

I

Nested column

You can easily put a whole new grade of columns inside an existing column. Now, that's called nesting. So, let's take a look at how that works. To nest columns, you simply create a new row inside an existing column. Now, this will create a new regular 12-column grid inside that existing column. And inside that column, you can use the same set of classes that you've been using so far. For example, we can create one main image with six sub-images as the following:



HTML

```
<div class="row">
     <section class="col-sm-8">
       <img src="images/image.jpg" alt="sample image">
       <h4>Exotic Pets</h4>
       <p>We offer <strong>specialized</strong> care for <em>reptiles,
       rodents, birds,</em> and other exotic pets.</p>
     </section>
     <section class="col-sm-4">
       <div class="row">
         <div class="col-2 col-sm-4">
            <img src="images/image.jpg" class="img-thumbnail"></div>
         <div class="col-2 col-sm-4">
            <img src="images/image.jpg" class="img-thumbnail"></div>
         <div class="col-2 col-sm-4">
            <img src="images/image.jpg" class="img-thumbnail"></div>
         <div class="col-2 col-sm-4">
            <img src="images/image.jpg" class="img-thumbnail"></div>
         <div class="col-2 col-sm-4">
            <img src="images/image.jpg" class="img-thumbnail"></div>
         <div class="col-2 col-sm-4">
            <img src="images/image.jpg" class="img-thumbnail"></div>
       </div> <!-- end of subnets row -->
     </section>
</div><!—end of row -->
```

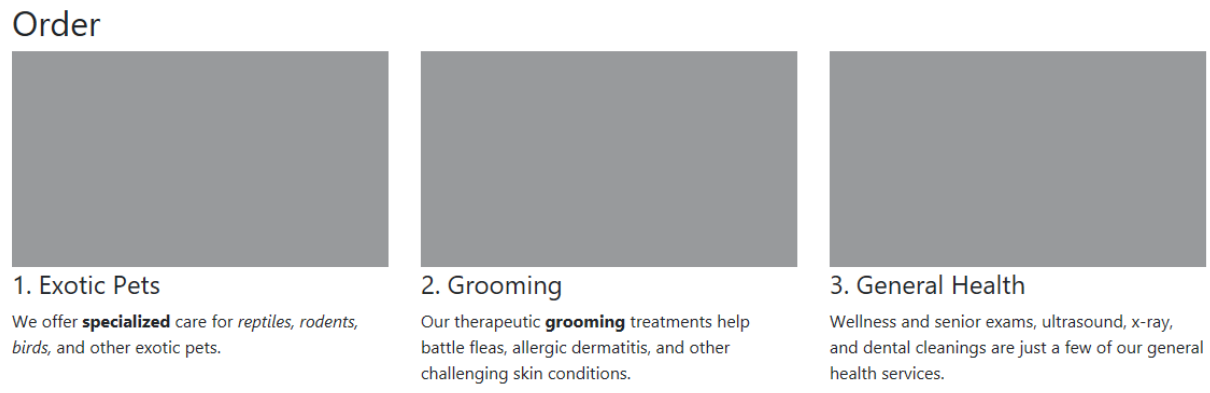
We can also set one main image with six sub-images below the main image:

Custom order

One of the more advanced ways that you can control layouts is to customize the order of elements. It helps you to reorder the way that columns appear and you can do this at different screen widths. Class ***col order-2*** change the order of the element in a row

For example, if we have a session with col

```
<section class="col">
```



If we add order-3 to the first session, that session will be moved to the third position

```
<section class="col order-3">
```

Order

2. Grooming

Our therapeutic **grooming** treatments help battle fleas, allergic dermatitis, and other challenging skin conditions.

3. General Health

Wellness and senior exams, ultrasound, x-ray, and dental cleanings are just a few of our general health services.

1. Exotic Pets

We offer **specialized** care for *reptiles, rodents, birds,* and other exotic pets.

Grid Alignment

Because Bootstrap uses Flexbox to control layouts, there are many new classes to handle how elements align into one another as well as their containers.

Display properties

There are other ways beside the grid to control how elements are positioned within Bootstrap, so let's take a look at what they are and how to use them. Now, first of all is position. Now, if you're familiar with CSS, these work exactly like the regular CSS position property. You can use three options. You can make an element fixed to the top of the screen, which means it would get removed from the flow of the document and put essentially at the top of the window or at the top of the view port, and of course fixed-bottom would place the element at the bottom.

There's another one called sticky-top, and that one will allow you to have an element stick to the top of the view port or the screen as you scroll in elements. That's a really useful effect, but it is not well-supported in a lot of browsers. So, that would be especially Internet Explorer and Edge, and it's only supported in the latest versions of Chrome as well.

There's also display properties. The display properties mimic what is possible with CSS and opens up Bootstrap to Flexbox classes that can help you with layout. So use a display class, you use the ***d*** and then optionally a breakpoint, and then the type of display property that you want.

We have the full range of display properties that you have normally available in CSS also in Bootstrap. So you can say ***d*** and the ***none*** or display none at a specific breakpoint.

If you have the code `d(-md)-none`, that would display nothing at the medium breakpoint. Bootstrap gives you a great way to control how your display properties are working at different breakpoints.

There are also a few classes that can help you control when something shows up if you use a paper printer. For example, you can say something like **`d-print-none`** when you want something not to print to paper. But you can also combine this with the regular display classes as well.

If you have the line,

`d-none d-print-block`

it only shows up when you're sending this element to your printer

Basic flex container

You start with the keyword d, and then you can optionally add one of the breakpoints so that the element will display as a Flex element only at certain breakpoints. That makes it super flexible. If you want to, you can use optionally a keyword called inline. By default, Flex elements are block-level elements, and so if you want to make it optionally an inline element, you can add this inline keyword. Of course, you also have to have the keyword flex.

One of the additional classes that we can use on the containers is the direction classes, and they look something like this.

Another thing that you can do with flexbox component is to control the order of the elements within the flexbox container. Now, the way that you do this is by using a class called ***`order`*** and then optionally specify a breakpoint and then a number from 1 through 12.

Furthermore, you can also specify justification for the items. Justifying content lets you control the spacing between items. Like other options, you can control them with an optional breakpoint. The alignment options are start, which puts the items to the left and the extra space to the right. End, which puts the items to the right and the extra space on the left.

Center, which is a great way to center content horizontally. Around is going to put the space around and in between elements. And between does something similar but it makes the edge element align to the edge. If you are familiar with how this work when we were working with columns, then this will be pretty much the same thing

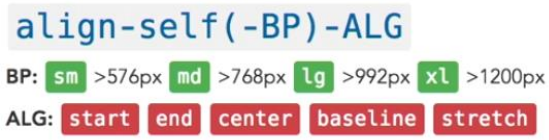
`wrap` lets you control whether the elements wrap in relation to the space in their container, and there's two options here.

You can either wrap the element or not wrap the element. Of course, you can include an extra keyword here to reverse the order of the elements and you also have the usual breakpoints.

Then finally, we have vertical alignment. Sometimes you want to align things vertically and so, you use this format here. You say align-content and then one of the breakpoints, and then finally, one of these alignment keywords.

Individual flex elements

Just like we can control a group of elements in Flex by controlling the container, we can also control individual Flex elements. And the way that that works is by using a class called ***`align-self.`***



Another thing that you can do in this new version of Bootstrap is to control the order of individual elements, as long as they are being displayed in Flexbox.

Floating element

Floating elements is a bit less popular now that we have Flexbox available to us, but there's still some instances where it might be necessary to work with it, so let's take a look at the classes that allow you to work with elements that are floating. Now they're pretty easy to use, and if you're used to the formulas, this is what this one looks like

Margin and padding

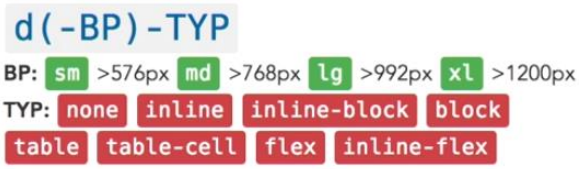
One of the richest and possibly most scary group of classes in Bootstrap are the spacing classes, which allow you control the margin and the padding of elements. Now, this mimics exactly the margin and padding classes in CSS. Once you get used to them though, you'll pick them up really quick, so let's take a look at the formula

Visibility

It's really common to show or hide content and you can do that easily with Bootstrap using the display classes, plus a couple of other classes that are related to screen readers.

There are two classes, one called `invisible` and the other one called `visible`. They don't really change the display property itself. `invisible` of course will make an element not appear on screen, but it will still take up the amount of room that it would normally take up.

Whereas *visible* means that it will be visible, but only to screen readers. So it's a way to force an element to show up with somebody with assistive devices. Now the other way to show or hide elements is simply using the *display* property, and here's the formula for that.



Sizing

Bootstrap gives you a few sizing utilities to control the width and height of different elements

Borders

There a number of classes that you can use to control how Bootstrap displays borders. The regular border class looks something like this

Navigation bar

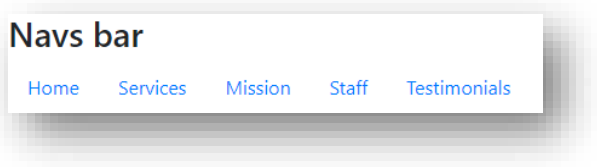
Every web project needs great navigation. Next to the grid, this is the most important component in Bootstrap. The different types of navigation components are navs, tabs, pills, and navbars. Navs are parents to all the other navigation. Tabs and pills are the same type of component and help you create content within a page that changes when clicked. Navbars, on the other hand, are typically used for main navigation between pages. Now, although you'll end up using navbars more often, it's easier to learn about navs first.

Creating Navigation Bar using *nav*

There's a group of components called **`nav`**. Now, this component had a lot of changes in this new version of Bootstrap due to the implementation of Flexbox and, thankfully, became a lot simpler to use. Now to create navs you can use either the unordered lists or use nav and regular divs.

Using unordered list, you will need to call the `nav` class in the <ul>, `nav-item` in each list <li>, and `nav-link` in each link.

```
 <ul class="nav">
        <li class="nav-item"><a href="#" class="nav-link">Home</a></li>
        <li class="nav-item"><a href="#" class="nav-link">Services</a></li>
        <li class="nav-item"><a href="#" class="nav-link">Mission</a></li>
        <li class="nav-item"><a href="#" class="nav-link">Staff</a></li>
        <liclass="nav-item"><a href="#" class="nav-
link">Testimonials</a></li>
      </ul>
```



If you want to you can add a couple of styles to these. You can add an **`active`** style. This style with shows when you use pills or nav style.

```
  <li class="nav-item"><a href="#" class="nav-link active">Home</a></li>
```

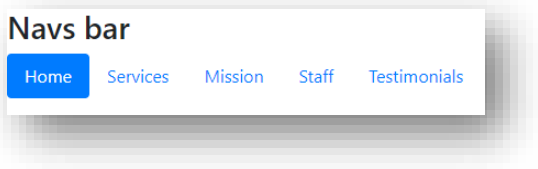
The other additional look that you can have for the links is **`disabled`**. Disabled basically disable the hyperlink

```
  <li class="nav-item"><a href="#" class="nav-link disabled">Home</a></li>
```

Nav-pills

We have nav-pills class looks like sort of buttons.

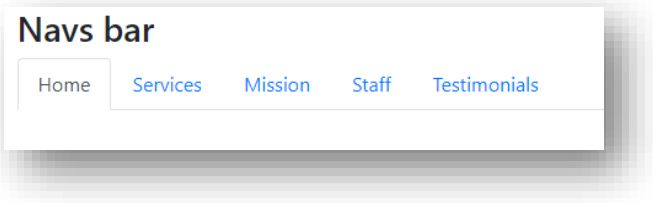
```
<ul class="nav nav-pills">
```



Nav-tabs

Nav-tabs class makes the link look like tabs

```
<ul class="nav nav-tabs">
```

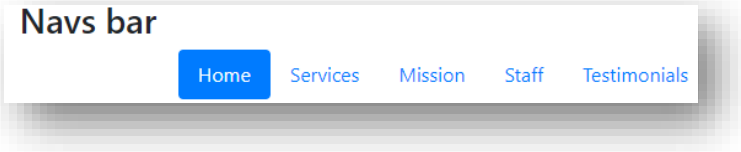


Creating navigation using divs. They look the same as using unordered list but simplier syntax.

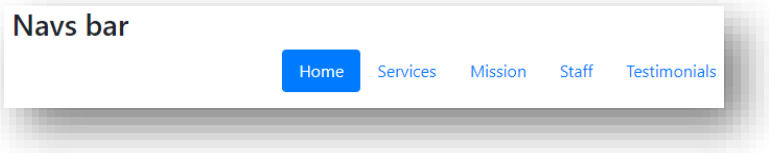
```
<div class="nav nav-tabs">
      <a href="#" class="nav-item nav-link active">Home</a>
      <a href="#" class="nav-item nav-link">Services</a>
      <a href="#" class="nav-item nav-link">Mission</a>
      <a href="#" class=" nav-item nav-link">Staff</a>
      <a href="#" class="nav-item nav-link">Testimonials</a>
    </div>
```

There's a couple of classes related to that. We can use justify-content-center to center the tabs, or justify-content-end to shift the tabs to the right side.

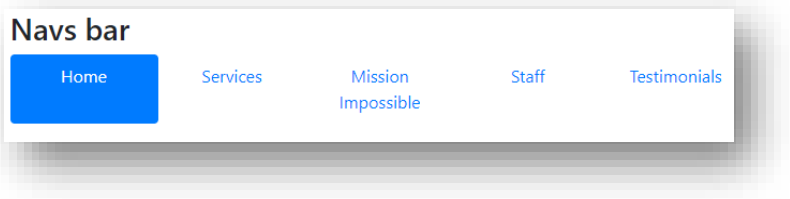
```
<div class="nav nav-pills justify-content-center">
```



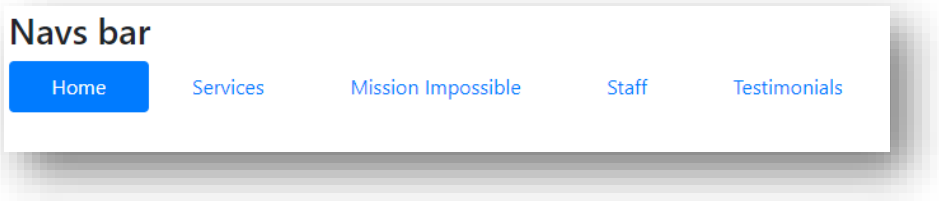

```
<div class="nav nav-pills justify-content-end">
```



you can also try `nav-justified` to make all the links of the same size.

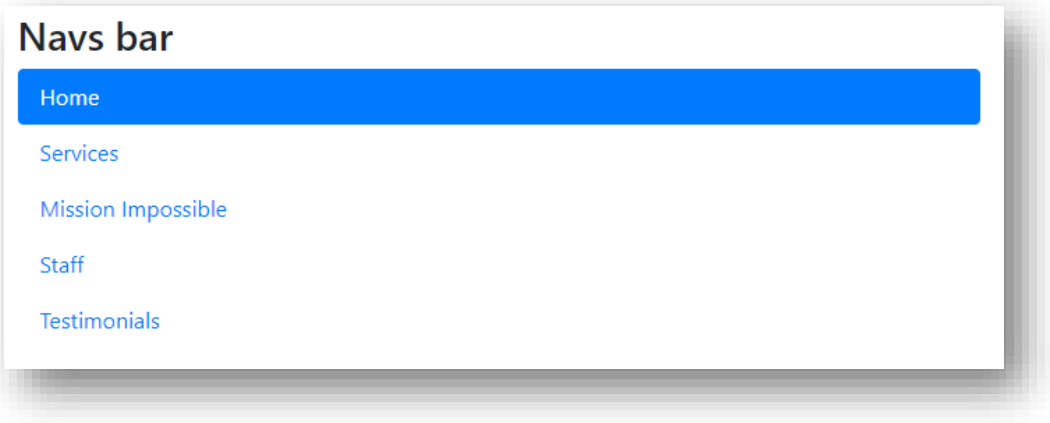


`nav-fills` makes all the links to fill into one line



Flex-column

`flex-column` makes all the links to automatically stack on top of one another.



you can actually specify a breakpoint at which these get converted back to rows. For example, stack all the links in medium size window.

```
<div class="nav nav-pills flex-column flex-md-row">
```

Using navbar classes

The navbar component is related to the nav component, so if you're familiar with navs, then creating navbars will be simple. the navbar class is the class that goes on the main container, and it can have a

number of other elements inside it. Now, by default, navbar components will stack, so we need to add a navbar-expand, with optional breakpoints to control when the navbar is going to expand.

```
  <nav class="navbar bg-dark navbar-dark">
        <div class='navbar-nav'> <!—div to store all the link →
        <a href="#" class="nav-item nav-link active">Home</a>
        <a href="#" class="nav-item nav-link">Services</a>
        <a href="#" class="nav-item nav-link">Mission Impossible</a>
        <a href="#" class=" nav-item nav-link">Staff</a>
        <a href="#" class="nav-item nav-link">Testimonials</a>
      </div>
      </nav>
```

Navs bar

by default, the navigation will always stack. Bootstrap is designed as a mobile first framework, so the default is to always do what would happen on a mobile device. So, we have an extra class that we need to add here, and it's called `navbar-expand` with a specific breakpoint.

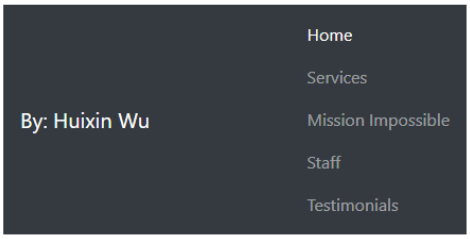
```
      <nav class="navbar bg-dark navbar-dark navbar-expand-sm">
```

Navbar brand and text

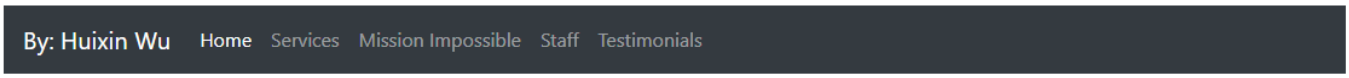
In addition to the `navbar nav` you can add other components to your navigation. The simplest to use is called the navbar brand and it's used for text or logos and will get an enhanced look. Another simple component is the navbar text component and you can add inline text that works well with the rest of the navigation. So, both of these are going to go right next to the navbar nav component. Navbar-brand is the class that you add when you want to add a logo or some text.

```
<div class='container'>
      <nav class="navbar bg-dark navbar-dark navbar-expand-sm">
        <!-- Create a div for navbar-brand -->
        <a class=navbar-brand href='#'>
            <img src='images/wisdompetlogo.svg' style="width: 40px;"
alt="wisdow pet logo"/>
        </a>
        <!-- nav list-->
        <div class='navbar-nav'>
          <a href="#" class="nav-item nav-link active">Home</a>
          <a href="#" class="nav-item nav-link">Services</a>
          <a href="#" class="nav-item nav-link">Mission Impossible</a>
          <a href="#" class=" nav-item nav-link">Staff</a>
          <a href="#" class="nav-item nav-link">Testimonials</a>
        </div>
      </nav>
    </div>
```

Navs bar

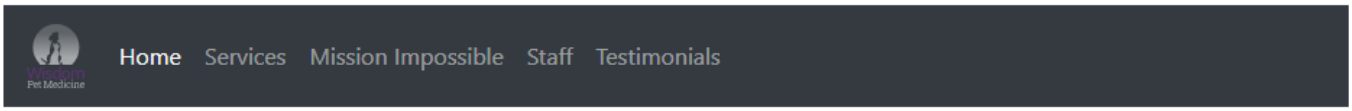


Navs bar



The `navbar-brand` can also work with link and/or image

```
<a class=navbar-brand href='#'>
            <img src='images/wisdompetlogo.svg' style="width: 40px;"
alt="wisdow pet logo"/>
 </a>
```

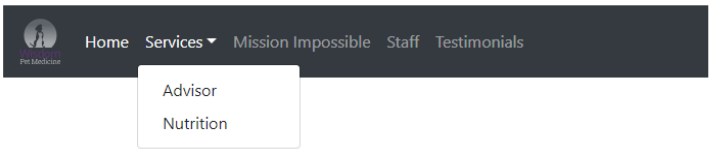


Dropdown

Although dropdowns are technically a separate component, they are used very often in menus. A dropdown requires a container, just like a lot of other elements to work. Therefore,, you'll need to create something that has a class of dropdown. This is usually a div, or it could be on a list item if you're using a navigation that uses list items.

```
 <div class='container'>
      <nav class="navbar bg-dark navbar-dark navbar-expand-sm">
        <!-- Create a div for navbar-brand -->
        <a class=navbar-brand href='#'>
            <img src='images/wisdompetlogo.svg' style="width: 40px;"
alt="wisdow pet logo"/>
        </a>
        <!-- nav list-->
        <div class='navbar-nav'>
          <a href="#" class="nav-item nav-link active">Home</a>
          <!-- specify that the dropdown list will be at Services-->
            <div class='dropdown'>
              <a class='nav-item nav-link dropdown-toggle' data-
toggle='dropdown' aria-haspopup='true' aria-expanded='false'
id="serviceDropdown"href='#'>Services</a>
              <!-- create a dropdown list for services-->
              <div class='dropdown-menu' aria-labelledby-'serviceDropdown'>
                  <a class="dropdown-item" href="#" >Advisor</a>
                  <a class="drowdown-item" href="#" >General Nutrition</a>
              </div> <!--end of drowdown list -->
```

```
          </div> <!-- end of service dropdown -->
        <a href="#" class="nav-item nav-link">Mission Impossible</a>
        <a href="#" class=" nav-item nav-link">Staff</a>
        <a href="#" class="nav-item nav-link">Testimonials</a>
      </div>
    </nav>
  </div>
```



Create collapsible content

Every so often, we need to create a navigation that collapses when the width of the device or the window is really small. That's because navigation can take up a lot of room vertically on a small device and often, we want to make sure that that room isn't taken over unless the user wants it. So in order to do that, we create these collapsible sections of content and to do that in Bootstrap, you essentially need two parts, the thing that you want to collapse and some element that will control that content.

So we need to work on two different parts. For the collapsible content, you're going to need a couple of classes, the first one is the generic collapse class that goes on the content that you of course want to collapse. And in addition to that, you're also going to use another class called navbar collapse because most of the time when you're using this feature, you're going to be collapsing a navbar. Now there are some other things that you need to do and we need to tie the element that we are collapsing which would be this element with the thing that is collapsing this element which will be a little button that's known as the hamburger menu button.

```
<nav class="navbar bg-dark navbar-dark navbar-expand-sm">
        <!-- Create a div for navbar-brand -->
        <a class=navbar-brand href='#'>
            <img src='images/wisdompetlogo.svg' style="width: 40px;" alt="wisdow pet
logo"/>
        </a>
        <!-- Create a hamburger button to control the collapseTab-->
        <button class='navbar-toggler' type='button' data-toggle='collapse' data-
target='#collapseTab' aria-controls="collapseTab" aria-expanded='false' aria-
label='Toggle naviagation'>
            <span class='navbar-toggler-icon'></span>
        </button> <!--end of collapse button -->

        <!-- Create a div to specified the collapse session -->
        <div class='collapse navbar-collapse'  id='collapseTab'>
        <!-- nav list-->
        <div class='navbar-nav'>
          <a href="#" class="nav-item nav-link active">Home</a>
          <a href="#" class="nav-item nav-link">Services</a>
          <a href="#" class="nav-item nav-link">Mission Impossible</a>
          <a href="#" class=" nav-item nav-link">Staff</a>
          <a href="#" class="nav-item nav-link">Testimonials</a>
        </div>
      </div> <!-- end of collapse-->
    </nav>
```

Styling elements

In this section, we're going to be working with some styles that can make your life easier but you probably won't need all the time. One of the most popular components are buttons. You use them primarily in forms and form components but there are so many options that they deserve their own study. Bootstrap styles are so well-made that even when I build a non-Bootstrap project, I'll often borrow CSS from Bootstrap projects. Bootstrap 4 adds something called tags which are a simple style for organizing content.

Progress bars are a good example of a style that might be really useful for some developers building interfaces that need to show progress. List groups are mobile friendly layouts to group content within list elements. That's something you're going to need often but especially useful for mobile projects. Finally, breadcrumbs are a common pattern for displaying channels within websites. So, before we get into more complex Bootstrap CSS

Creating buttons

Now there are a number of classes that help you create buttons.

- `btn` basic class
- `btn-SIZ`
  SIZ: `sm` `lg`
- `<a>` `<button>` `<input>`

Now you do have to specify at least one of the contextual color classes. So you would say **`btn`** and then hyphen and then one of these colors that you see right here

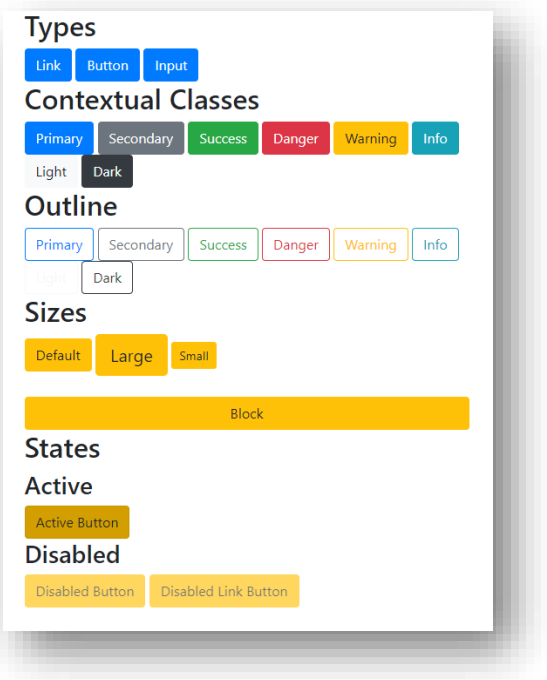
There is an alternative for outlines which looks like this, you say **`btn`** and then outline and then one of the contextual color classes that you see right here.

In addition to that there's a special style called **`btn-block`** that you can add if you want a button to occupy the entire width of the container, and in addition to that you have the traditional active, as well as disable classes that you can use.

- `btn-block` full width
- `active`
- `disabled`

Example)

```
<h2>Types</h2>
<a class='btn btn-primary' href="#" role="button">Link</a>
<button class='btn btn-primary' type="submit">Button</button>
<input class='btn btn-primary' type="button" value="Input">

<h2>Contextual Classes</h2>
<button class='btn btn-primary'>Primary</button>
<button class='btn btn-secondary'>Secondary</button>
<button class='btn btn-success'>Success</button>
<button class='btn btn-danger'>Danger</button>
<button class='btn btn-warning'>Warning</button>
<button class='btn btn-info'>Info</button>
<button class='btn btn-light'>Light</button>
<button class='btn btn-dark'>Dark</button>

<h2>Outline</h2>
<button class='btn btn-outline-primary'>Primary</button>
<button class='btn btn-outline-secondary'>Secondary</button>
<button class='btn btn-outline-success'>Success</button>
<button class='btn btn-outline-danger'>Danger</button>
<button class='btn btn-outline-warning'>Warning</button>
<button class='btn btn-outline-info'>Info</button>
<button class='btn btn-outline-light'>Light</button>
<button class='btn btn-outline-dark'>Dark</button>

<h2>Sizes</h2>
<button class='btn btn-warning'>Default</button>
<button class='btn btn-warning btn-lg'>Large</button>
<button class='btn btn-warning btn-sm'>Small</button>
<br/>
<br/>
<button class='btn btn-warning btn-block'>Block</button>

<h2>States</h2>
<h3>Active</h3>
<button class='btn btn-warning active' >Active Button</button>

<h3>Disabled</h3>
<button class='btn btn-warning disabled'>Disabled Button</button>
<a class='btn btn-warning disabled' href="#">Disabled Link Button</a>
```



Grouping buttons

There are a few classes that help you create button groups. To group simple buttons together you can use the **`btn-group`** class. There's also a vertical option for button groups that stack on top of one another. Those are traditionally used for mobile devices but you can use them for whatever you want. Finally, there is a toolbar option that let's you create groups of button groups. Now when you're creating button groups it's a good idea to use the aria label property to add a little bit of context to your button groups.

- `btn-group`
- `btn-group-vertical`
- `btn-toolbar`

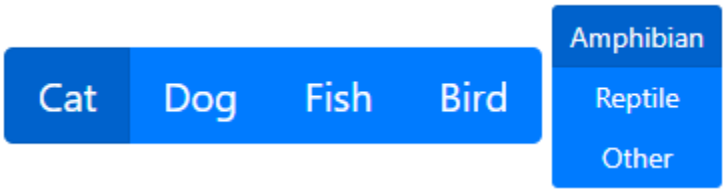
Example)

```
<h1>Our Mission</h1>
<div class='btn-group'>
    <button type="button" class="btn btn-primary active">Cat</button>
    <button type="button" class="btn btn-primary">Dog</button>
    <button type="button" class="btn btn-primary">Fish</button>
    <button type="button" class="btn btn-primary">Bird</button>
</div>
<div class='btn-group'>
    <button type="button" class="btn btn-primary active">Amphibian</button>
    <button type="button" class="btn btn-primary">Reptile</button>
    <button type="button" class="btn btn-primary">Other</button>
  </div>
```


```
<div class='btn-group-vertical'>
    <button type="button" class="btn btn-primary active">Amphibian</button>
    <button type="button" class="btn btn-primary">Reptile</button>
    <button type="button" class="btn btn-primary">Other</button>
  </div>
```

You can also add button sizes:

```
<div class='btn-group btn-group-lg'>
<div class='btn-group-vertical btn-group-sm'>
```



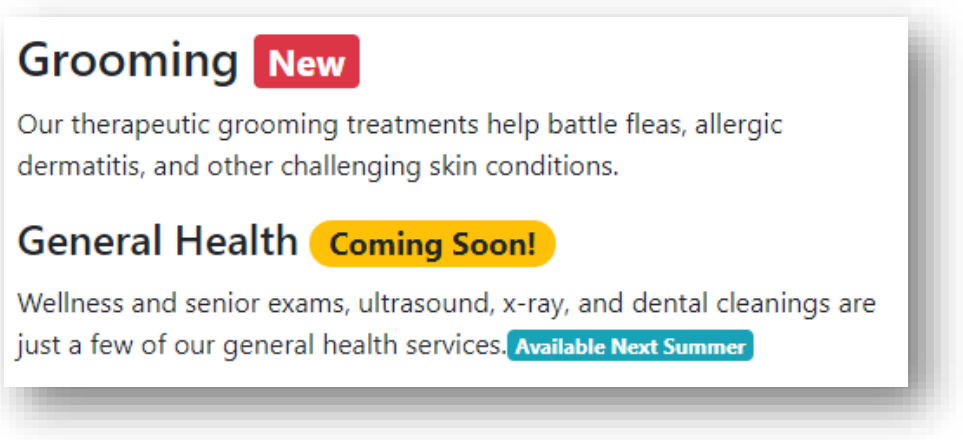
Badges

Badges are usually created using the element , the main class for badges is called **badge**, and then, in addition to that, you have a slightly different shape for the badges, called **badge-pill**. You can also use the traditional contextual classes. One thing about badges is that they are contextual, so they will resize according to the container.

Example)

```
<h3>Grooming <span class='badge badge-danger'>New</span></h3>

<p>Our therapeutic <span>grooming</span> treatments help battle fleas,
allergic dermatitis, and other challenging skin conditions.</p>


<h4>General Health <span class='badge badge-warning badge-pill'>Coming
Soon!</span></h4>
```

```
<p>Wellness and senior exams, ultrasound, x-ray, and dental cleanings are
just a few of our general health services.<span class='badge badge-
info'>Available Next Summer</span></p>
```



Progress bars

If you need to create progress bars, Bootstrap makes it easy with a number of classes to take care of them. In order to do so, you need two sets of tags. The main container is the progress container, and then internally, you create a container with the progress-bar class. To control the width and height of the progress bar, you can either use the width classes, which are the w-25, 50, 75 or 100, or directly use the style attribute to control the width and the height. You can also add a label by putting in some text between the internal progress bar element

- `progress` containers
- `progress-bar` item
- Style `width`, `height`
- Label text

You also have color styles. The bars are blue by default or the primary color. If you want to, you can use one of the contextual color classes to change the color of the context of the bar. Now there is an alternative style that you can use called progress-bar-striped, and that gives you a bar that has some striped sort of bars on it. If you want to, you can animate that with progress-bar-animated, and you can use as many bars as you want by adding multiple progress bar components inside the main progress bar element. So all you do is, you just add a different progress bar item for every sort of progress bar that you want in kind of the main component.

- Use `bg-COLOR`

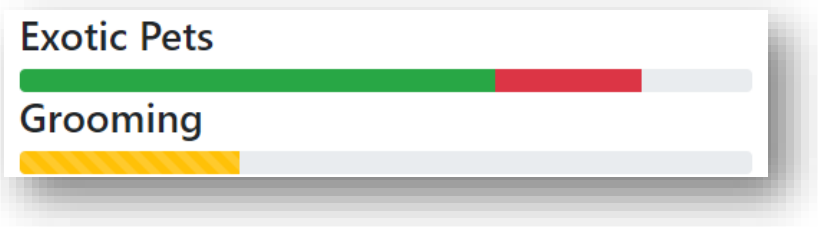
  `success` `info` `warning` `danger`
- `progress-bar-striped`
- `progress-bar-animated`
- Multiple bars

You should use a series of tags to handle accessibility like **`role="progressbar"`** to identify this as a progress bar. You can also use **`aria-valuenow`** to store the current value of the bar, as well as **`aria-valuemin`** to hold a minimum value of the element, and **`aria-valuemax`** to hold the maximum value of the element. This is going to make your progress bars a little more compatible with screen readers.

- `role="progressbar"`
- `aria-valuenow`
- `aria-valuemin`
- `aria-valuemax`

Example)

```
<h3>Exotic Pets</h3>
<!-- progress bars-->
  <div class='progress'>
    <div class='progress-bar bg-success' style='width:65%;'></div>
      <div class='progress-bar bg-danger' style='width:20%;'></div>
  </div> <!-- end of progress bar -->

<h3>Grooming</h3>
<!-- progress bars-->
  <div class='progress'>
          <div class='progress-bar bg-warning progress-bar-striped
          progress-bar-animated' style='width:30%;'></div>
  </div> <!-- end of progress bar -->
```



Predesign layout components

Some styles in Bootstrap are designed to help you quickly create elements for certain types of content. These are sometimes called design patterns and Bootstrap has some really useful patterns for common layout tasks. Design patterns are predefined through various components. A jumbotron, for example, is a popular way to display some content at the top of a website. Table styles are extremely useful because they make elements that are challenging to style look great without a lot of work. Card layouts are a new design pattern that is available in this version of Bootstrap.

They let you create designs that are boxed into items that look like cards. Another really useful design pattern is media elements. They're great for creating lists of items that need to have photos on one side and headlines and texts in another.

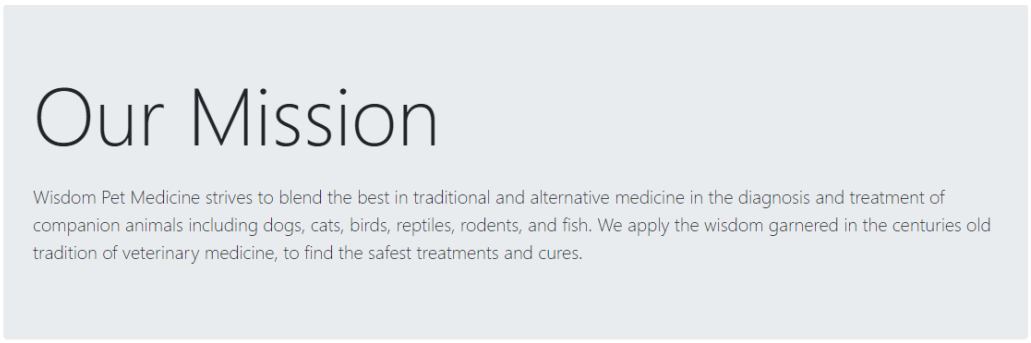
Jumbotron

The jumbotron is a component for a highlighted section of content and it's commonly used at the top of many websites. It's really easy to do with Bootstrap and this current version has a couple of options for you.

- `jumbotron` container
- `jumbotron-fluid` items
- Use styles as needed

Example)

```
<header class='jumbotron'>

    <div class="display-2 mb-4">Our Mission</div>

    <p class="lead">Wisdom Pet Medicine strives to blend the best in
traditional and alternative medicine in the diagnosis and treatment of
companion animals including dogs, cats, birds, reptiles, rodents, and fish.
We apply the wisdom garnered in the centuries old tradition of veterinary
medicine, to find the safest treatments and cures.</p>

</header>
```




```
<header class='jumbotron jumbotron-fluid'>

    <div class='container'>

      <div class="display-2 mb-4">Our Mission</div>

      <p class="lead">Wisdom Pet Medicine strives to blend the best in
traditional and alternative medicine in the diagnosis and treatment of
companion animals including dogs, cats, birds, reptiles, rodents, and fish.
We apply the wisdom garnered in the centuries old tradition of veterinary
medicine, to find the safest treatments and cures.</p>

    </div>

  </header>
```

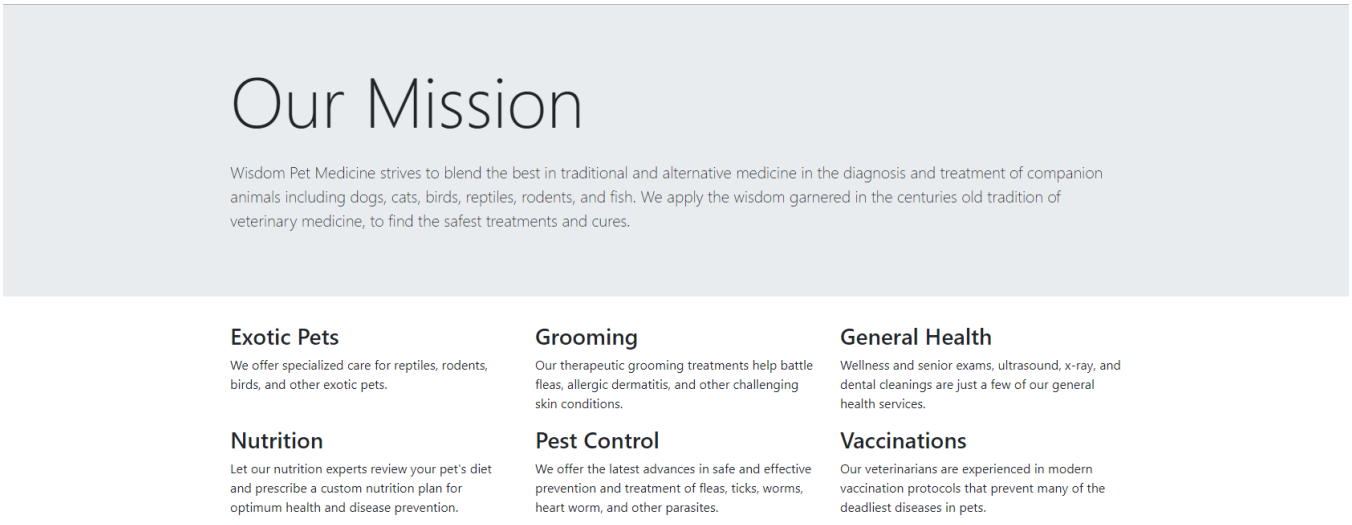


Table style

One of the best features in Bootstrap is how you control the look and feel of tables. Now there is a couple of main classes that you can use to do this and the table class is something that goes on the table tag itself. When you do that, Bootstrap will immediately take over and give you a nicely rendered and great looking table. Now there is an option that you can add as a separate class called table dark if you want essentially what's an inversed table which has a black background and light text.

In addition to that, you have a few style classes that you can use. So if you add a class of table striped, it's going to change the color of every other cell to give you a stripped table. You can also ask the table to have a border with the table bordered class. And if you include a table hover class, when you move your mouse on top of one of the cells, it'll highlight the entire row.

- `table-striped`
- `table-bordered`
- `table-hover`

There's a number a ways of adding colors. In the head section of a table you can add a thead light or a thead dark class and that will make that entire row have a dark background or a light background.

You also have the ability of controlling the size of the table if you include a table sm on the table tag itself, it's going to make a table that has a little less padding to it.

Example)

Using the class table in table

```
<table class='table'>
```

it will add certain alignment and style to a table

Item #	Product or Service	Price (ea.)	Retail Price (Case)	Case Discount	Wholesale Price	Wholesale Discount
100050	Advance Pet Oral Care Toothbrush and Toothpaste	$9.55	$108.87	$5.73	$103.14	$11.46
100043	Basic Teeth Cleaning and Exam	$100.00	$1,140.00	$60.00	$1,080.00	$120.00

Table-striped makes background-light color every other rows.

```
<table class='table table-striped'>
```

Item #	Product or Service	Price (ea.)	Retail Price (Case)	Case Discount	Wholesale Price	Wholesale Discount
100050	Advance Pet Oral Care Toothbrush and Toothpaste	$9.55	$108.87	$5.73	$103.14	$11.46
100043	Basic Teeth Cleaning and Exam	$100.00	$1,140.00	$60.00	$1,080.00	$120.00
100013	Calm Cat Anxiety Relief Spray	$9.49	$108.19	$5.69	$102.49	$11.39

Table-hover turns the row background-light color on hover

```
<table class='table table-hover'>
```

Table-border border the entire table

```
<table class='table table-bordered'>
```

Item #	Product or Service	Price (ea.)	Retail Price (Case)	Case Discount	Wholesale Price	Wholesale Discount
100050	Advance Pet Oral Care Toothbrush and Toothpaste	$9.55	$108.87	$5.73	$103.14	$11.46
100043	Basic Teeth Cleaning and Exam	$100.00	$1,140.00	$60.00	$1,080.00	$120.00

`table-sm` turns off all the small spaces within your cell by getting off of some of the padding

Item #	Product or Service	Price (ea.)	Retail Price (Case)	Case Discount	Wholesale Price	Wholesale Discount
100050	Advance Pet Oral Care Toothbrush and Toothpaste	$9.55	$108.87	$5.73	$103.14	$11.46
100043	Basic Teeth Cleaning and Exam	$100.00	$1,140.00	$60.00	$1,080.00	$120.00
100013	Calm Cat Anxiety Relief Spray	$9.49	$108.19	$5.69	$102.49	$11.39
100041	Cat Hairball Remedy Gel	$6.00	$68.40	$3.60	$64.80	$7.20

you can also add individual color to individual table rows or cells

```
 <tr><th scope="row">100050</th><td class='table-danger'>Advance Pet Oral Care
Toothbrush and Toothpaste</td><td>$9.55
</td><td>$108.87</td><td>$5.73</td><td>$103.14</td><td>$11.46</td></tr>

      <tr class='table-primary'><th scope="row">100043</th><td>Basic Teeth
Cleaning and Exam</td><td>$100.00
</td><td>$1,140.00</td><td>$60.00</td><td>$1,080.00</td><td>$120.00</td></tr>
```

Item #	Product or Service	Price (ea.)	Retail Price (Case)	Case Discount	Wholesale Price	Wholesale Discount
100050	Advance Pet Oral Care Toothbrush and Toothpaste	$9.55	$108.87	$5.73	$103.14	$11.46
100043	Basic Teeth Cleaning and Exam	$100.00	$1,140.00	$60.00	$1,080.00	$120.00

You can also add color to table head

```
<thead class='thead-light'><tr><th scope="col">Item #</th><th
scope="col">Product or Service</th><th scope="col">Price (ea.)</th><th
scope="col">Retail Price (Case)</th><th scope="col">Case Discount</th><th
scope="col">Wholesale Price</th><th scope="col">Wholesale
Discount</th></tr></thead>
```

Table-responsive

You can just say table responsive and what'll happen is if the table has too much info then you'll be able to scroll it horizontally. Optionally you can institute a break point at which that happens so you can say like table responsive and then this will only happen at the medium break point. But I'm not really sure that is super useful in this case because we don't have that much data so in some instances you may want to control that by break point.

But really I like the table responsive class. It is pretty useful and I think that when you have a lot of data horizontally, that can come in quite handy. So tables is one of those features I really love about bootstrap. Styling tables can be quite challenging. There's a lot of different classes that you have to take care of and bootstrap is essentially a framework and provides a framework for dealing with anything you would want to do with tables.

Cards

Cards are a new design element in Bootstrap 4 and it can help you lay out your content into card-like containers that look great and have as a small outline around them. There are a ton of options in terms of the classes that you can use.

You can modify the card content:

- `card-text`
- `card-title`
- `card-subtitle`
- `card-link`
- `card-img`

```
<section class="card mb-5" id="drwinthrop">
    <div class='card-body'> <!-- add a container to card elements -->
      <img class="card-img img-fluid" src="images/doctor-winthrop.jpg"
alt="Doctor Winthrop Photo">
    <h2 class='card-title'>Dr. Stanley Winthrop</h2>
    <h5 class='card-subtitle'>Behaviorist</h5>
      <p class='card-text'>Dr. Winthrop is the guardian of Missy, a three-
      year old Llaso mix, who he adopted at the shelter. Dr. Winthrop is
      passionate about spay and neuter and pet adoption, and works tireless
      hours outside the clinic, performing free spay and neuter surgeries for
      the shelter.</p>
    <a class='card-link' href="#">About Me</a>
    <a class='card-link' href="#">My Pets</a>
    <a class='card-link' href="#">Client Slideshow</a>
  </div>
  </section>
```

You can also add colors

Card content classes

The basic container should have a card class, and we should at least have an element that has a card body as a container of some of the content in your cards.

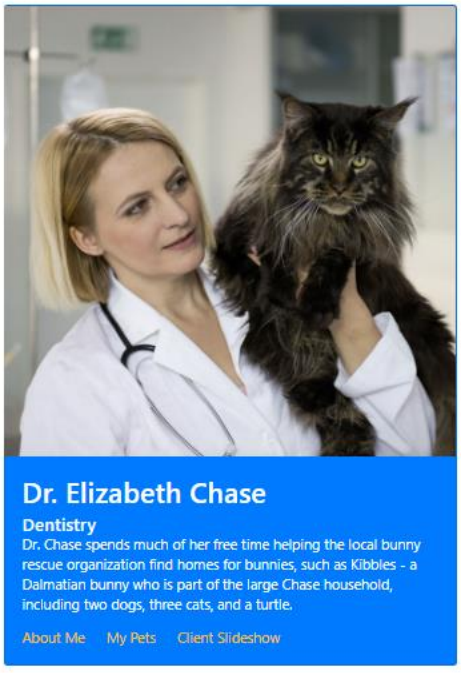
You can create a card header and footer by using classes card-header and card-footer

```
<section class="card mb-5 bg-light" id="drwinthrop">
    <div class='card-header'> <!-- card header-->
      <h2 class='card-title'>Dr. Stanley Winthrop</h2>
      <h5 class='card-subtitle'>Behaviorist</h5>
    </div>
    <div class='card-body'> <!-- add a container to card elements -->
    <img class="card-img img-fluid" src="images/doctor-winthrop.jpg"
alt="Doctor Winthrop Photo">
      <p class='card-text'>Dr. Winthrop is the guardian of Missy, a three-
      year old Llaso mix, who he adopted at the shelter. Dr. Winthrop is
      passionate about spay and neuter and pet adoption, and works tireless
      hours outside the clinic, performing free spay and neuter surgeries for
      the shelter.</p>
    <div class='card-footer'>  <!-- card footer-->
      <a class='card-link' href="#">About Me</a>
      <a class='card-link' href="#">My Pets</a>
      <a class='card-link' href="#">Client Slideshow</a>
    </div>
  </div>
  </section>
```

You can also work with images in a card

- `card-img`
- `card-img-top`
- `card-img-bottom`
- `card-img-overlay`

```
<section class="card mb-5 bg-primary text-white" id="drchase" >
      <img class="card-img-top img-fluid" src="images/doctor-chase.jpg"
alt="Doctor Chase Photo">
      <div class='card-body'>
          <h2 class='card-title'>Dr. Elizabeth Chase</h2>
          <h5 class='card-subtitle'>Dentistry</h5>
            <p class='card-text'>Dr. Chase spends much of her free time
            helping the local bunny rescue organization find homes for
            bunnies, such as Kibbles - a Dalmatian bunny who is part of the
            large Chase household, including two dogs, three cats, and a
            turtle.</p>
          <a class='card-link text-warning' href="#">About Me</a>
          <a class='card-link text-warning' href="#">My Pets</a>
          <a class='card-link text-warning' href="#">Client Slideshow</a>
        </div>
  </section>
```

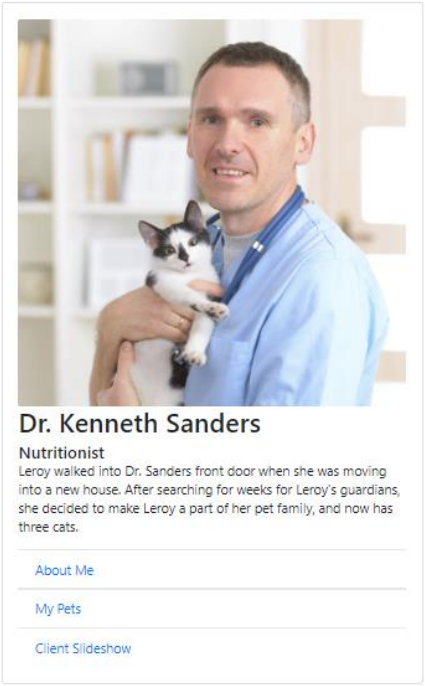


Normal list group container class that will go around the elements, and then inside that, each of the items that is part of the list gets a list-group-item. Finally there is a class called list-group-flush.

That allows you to take the content that is in the list group, and extend it to the edge of the card.

- `list-group` container
- `list-group-item`
- `list-group-flush`

```
<section class="card mb-5" id="drsanders">
    <div class='card-body'>
    <img class="card-img img-fluid" src="images/doctor-sanders.jpg"
alt="Doctor Sanders Photo">
      <h2 class='card-title'>Dr. Kenneth Sanders</h2>
      <h5 class='card-subtitle'>Nutritionist</h5>
            <p class='card-text'>Leroy walked into Dr. Sanders front door
            when she was moving into a new house. After searching for weeks
            for Leroy's guardians, she decided to make Leroy a part of her
            pet family, and now has three cats.</p>
      <div class='list-group list-group-flush'>
          <a class='list-group-item' href="#">About Me</a>
          <a class='list-group-item' href="#">My Pets</a>
          <a class='list-group-item' href="#">Client Slideshow</a>
        </div>
    </div>
  </section>
```

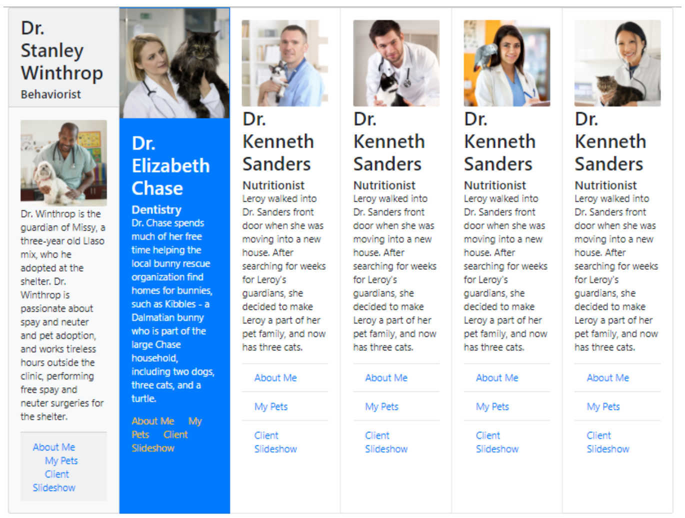


Card layout

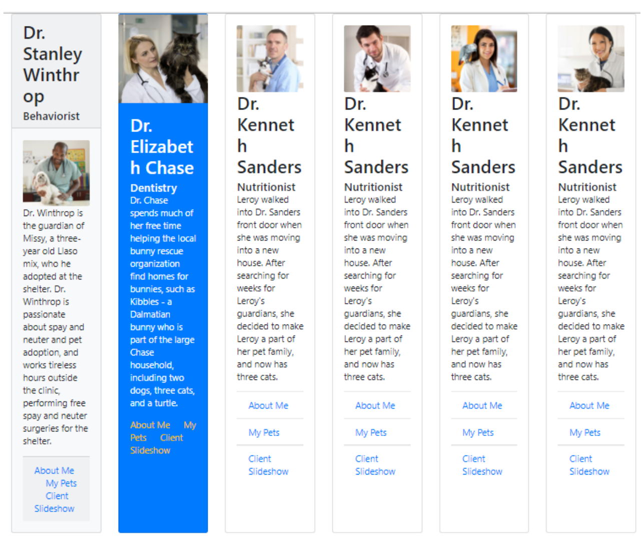
So, let's take a look at how we control the layout of different cards. And what you do in Bootstrap is you create a container that is going to have a number of cards in it and that container should get some classes. And the first one I'll talk about is called card group. Card group will place the cards side by side, and they will have a shared order. If you don't want a shared order, you can use card deck. It's pretty much the same thing as card group, puts everything side by side, but it does have a little of a spacing in between the cards.

And finally, there is something called card columns, this makes a really interesting layout where cards align. If you don't like any of these, you can use traditional rows and columns.

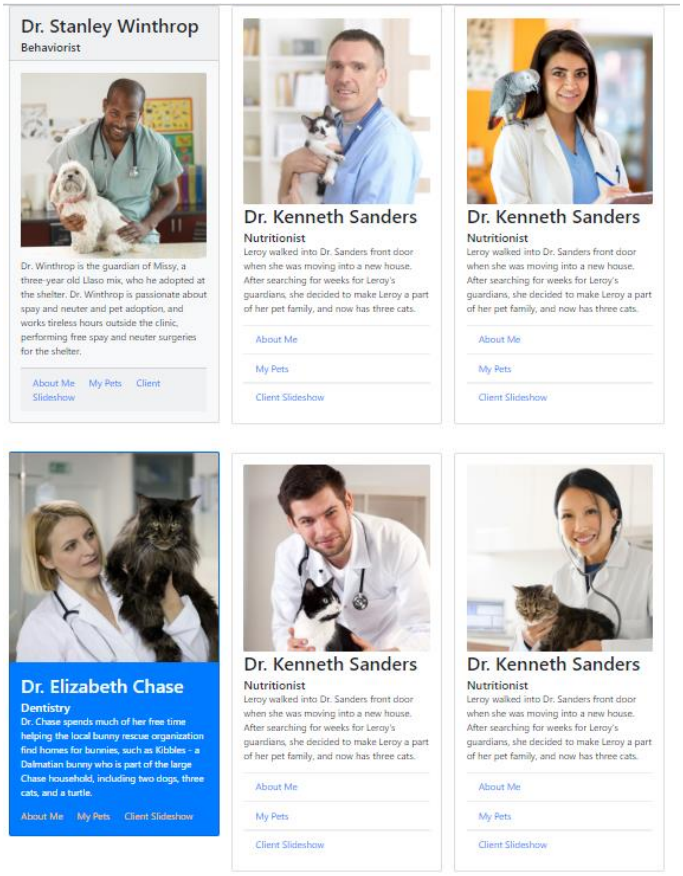
```
<div class="card-group">
```




```
<div class="card-deck">
```



```
<div class="card-columns">
```



Use can also use traditional row and col

```
<div class='container'>
  <div class='row'>
    <div class='col-lg-6'>
      <section class="card mb-5 bg-light" id="drwinthrop">
      </section>
    </div>
    <div class='col-lg-6'>
      <section class="card mb-5 bg-primary text-white" id="drchase" >
    </div>
    <div class='col-lg-6'>
      <section class="card mb-5" id="drsanders">
    </div>
    <div class='col-lg-6'>
      <section class=" card mb-5" id="drgardner">
      </div>
    <div class='col-lg-6'>
      <section class="card mb-5" id="drsanders">
    </div>
    </div><!--end of card-row -->
</div><!-- content container -->
```

Working with interactive components

Popovers

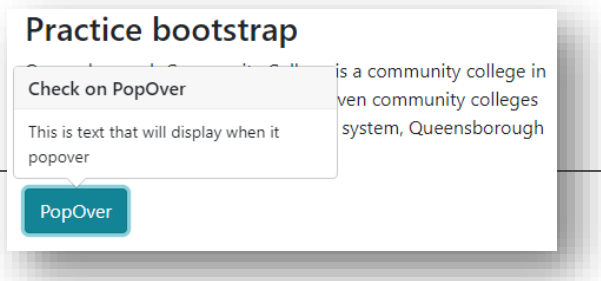
PopOvers are another component that lets you display additional content and are triggered by events like clicks. Now they're almost exactly like tool tips so if you're coming from that video, this is going to be pretty similar but they have a slightly different style. In order to set these up, you simply add a data-toggle attribute of popover to an element, and then you can add a title attribute with some text to it as well as a data-content where you add some additional content.

So the text would become like, the headline of the popover, and then the data content would be the sub-content underneath that, because they do have more of a headline section in them. Now in order to get these started, you do need to activate them, and you have to do that through jQuery so you use the jQuery, sort of start-up function, and then somewhere in there, you can activate all of the popups on the current page by targeting the data-toggle=popover attribute, and then you use the popover method right here, and you can optionally pass along some of the setup options.

```
$(function () {
  $('[data-toggle="popover"]').popover({ OPTIONS })
})
```

Example)

```
<div>
    <p>Queensborough Community College is a community college in
      <a href='#' data-toggle='tooltip' data-placement='top' title='You place
over here what you want to show up'>Bayside, Queens, New York.</a> One of
seven community colleges within the City University of New York system,
Queensborough enrolls more than 15,000 students.</p>
  </div>

  <!-- create a button to work with popover -->
    <button type='button' class='btn btn-info' data-toggle='popover' data-
trigger='hover' title='Check on PopOver' data-content='This is text that will
display when it popover'> PopOver</button>
<!-- jQuery first, then Popper.js, then Bootstrap JS -->
<script src="js/jquery.slim.min.js"></script>
<script src="js/popper.min.js"></script>
<script src="js/bootstrap.min.js"></script>
<!-- script toggle for tooltips-->
<script>
$(function(){
  $('[data-toggle="tooltip"]').tooltip();
  $('[data-toggle="popover"]').popover({placement:"top"});
});
</script>
```



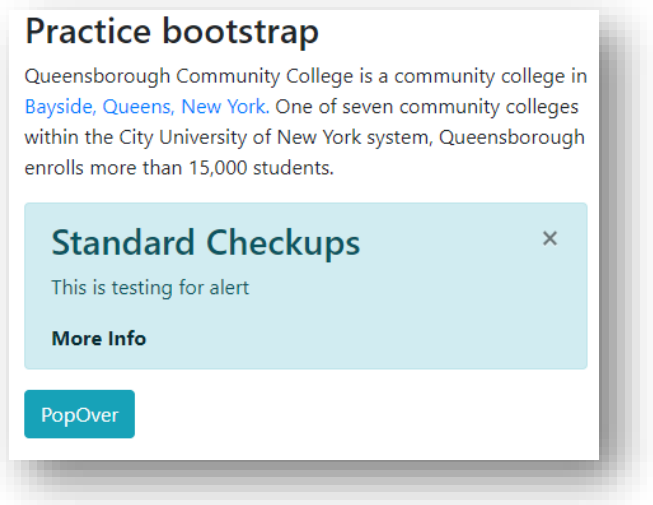
Setting alerts

Alerts are messages with special contextual styles that are designed to display in great looking boxes. They're optionally dismissed and can have any sort of markup so let's take a look at them. Now setting up alerts is pretty simply. You start out with the basic alert class on a container. There's a few contextual classes that you can use for color. So for example, success, info, warning, and danger. A few classes are available for internal content, like headings as well as links.

Although you can fit just about any HTML content inside an alert. Now just be aware that it is really meant for very simple content, so don't try too many crazy HTML tags in there. You can choose to dismiss the alerts so that the user can get rid of message. This is how you do that. You add a class of alert-dismissible and then if you want some animation you can also add the fade and show classes. So when you dismiss the alert, it'll have a little animation that fades out.

When you do this, you should also add a close button to the alert and here's what that code looks like.

```
  <!--create a alert text -->
  <div class='alert alert-info'>
    <button type='button' class='close' data-
dismiss='alert'><span>&times;</span>
    </button>
    <h3 class='alert-heading'>Standard Checkups</h3>
    <p> This is testing for alert</p>
    <a class='alert-link' href="#">More Info</a>
  </div>
```



Dropdown

- `dropdown`
- `dropdown-toggle`
- `dropdown-menu`
- `dropdown-item`

Dropdowns are a common design patterns in Bootstrap. You can create them easily for different components, so let's take a look. You can use dropdown in a bunch of different components including navs, tabs, and buttons. If you've been following along, you've already seen how to work them into navbars which is also how you implement them in navs. You create dropdowns in two parts. First, you create a button to trigger the dropdown and then create the menu itself. They're tied under our dropdown container that toggles the dropdown menu.

For the menu, you can either use anchor tag links or buttons. The classes for a basic dropdown are the dropdown class on the container of both the trigger and the menu element, the dropdown-toggle class on the button that triggers the menu, and then the **`dropdown-menu`** class which is the container for the menu. You also add a **`dropdown-item`** on each individual menu item which is either a link or a button. Inside the dropdown, you can add a few elements including at header that's a nice title to a group of elements in the dropdown, you can also you use a divider that lets you create a horizontal line to divide groups of elements, any menu item can also be tagged as disabled to prevent people from clicking on it.

- `dropdown-header`
- `dropdown-divider`
- `disabled`

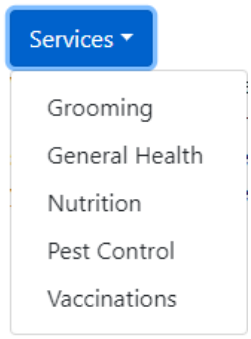
There's a few options that control how things look and work. For example, you can control the size of the buttons using the **`btn-sm`** for a smaller button and the **`btn-lg`** for a bigger button that goes in the trigger of the dropdown menu. You can also cause the button to drop upwards instead of downwards with the dropup class. If you want to align the menu to the right instead of to the left which is a default, you can add a dropdown-menu-right class. Finally, you can create a menu elements where the dropdown and button are split. That's a little bit more flexible than just having a dropdown since it allows you to choose an option before you submit the button.

- `btn-sm` `btn-lg`
- `dropup`
- `dropdown-menu-right`
- `btn-group` `dropdown-toggle-split`

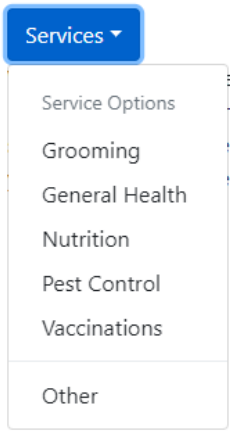
Example)

```
<div class='dropdown text-dark'>
```

```
<button type="button" id='dropdownMenuButton' class='btn btn-primary
dropdown-toggle ' data-toggle='dropdown'>Services</button>
  <div class='dropdown-menu' aria-labelledby='dropdownMenuButton'>
      <a class='dropdown-item' href="#">Grooming</a>
      <a class='dropdown-item' href="#">General Health</a>
      <a class='dropdown-item' href="#">Nutrition</a>
      <a class='dropdown-item' href="#">Pest Control</a>
      <a class='dropdown-item' href="#">Vaccinations</a>
  </div>
</div>
```




```
<div class='dropdown-menu' aria-labelledby='dropdownMenuButton'>
        <div class='dropdown-header'>Service Options</div>
            <a class='dropdown-item' href="#">Grooming</a>
            <a class='dropdown-item' href="#">General Health</a>
            <a class='dropdown-item' href="#">Nutrition</a>
            <a class='dropdown-item' href="#">Pest Control</a>
            <a class='dropdown-item' href="#">Vaccinations</a>
        <div class='dropdown-divider'></div>
            <a class='dropdown-item' href='#'>Other</a>
    </div>
```



Add tooltips

Tooltips are a great way to show some simple additional information on links and buttons. Now you should only use tooltips on HTML elements that are focusable like links and foreign controls because of

problems with usability. In order to set these up, you add a data-toggle of tooltip to an element and then add a title attribute with some text in it.

Tooltips are configured in one of two ways, you can use a data attribute or JavaScript. Data is a lot easier and it's what I'll be showing you here, however you can do anything that you can with the data attributes in JavaScript as well. Tooltips are not turned on by default so if you do want to use them, you will still need to use JavaScript, even if you're using data attributes. Now if you do want to configure them with JavaScript, you would use something like this.

```
$(function () {
  $('[data-toggle="tooltip"]').tooltip({ OPTIONS })
})
```

```
<div class="tooltip bs-tooltip-top" role="tooltip">
  <div class="arrow"></div>
  <div class="tooltip-inner">
    Some tooltip text!
  </div>
</div>
```

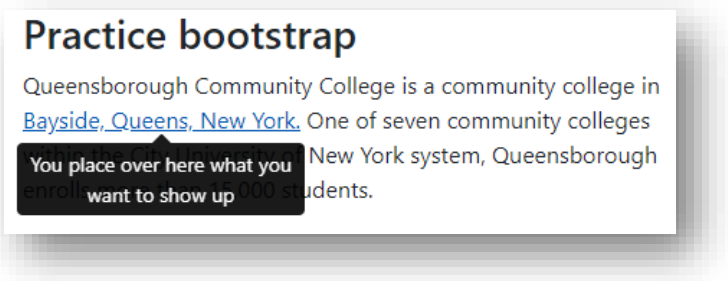
There are different options that you can add to the tooltips. Some of the basic options that you can add to the tooltips are:

placement top right bottom left

trigger click hover focus

Example)

```
<div>
    <p>Queensborough Community College is a community college in
      <a href='#' data-toggle='tooltip' data-placement='top' title='You place
      over here what you want to show up'>Bayside, Queens, New York.</a> One
      of seven community colleges within the City University of New York
      system, Queensborough enrolls more than 15,000 students.</p>
  </div>
<!-- jQuery first, then Popper.js, then Bootstrap JS -->
<script src="js/jquery.slim.min.js"></script>
<script src="js/popper.min.js"></script>
<script src="js/bootstrap.min.js"></script>
<!-- script toggle for tooltips-->
<script>
$(function(){
  $('[data-toggle="tooltip"]').tooltip();
});
</script>
```



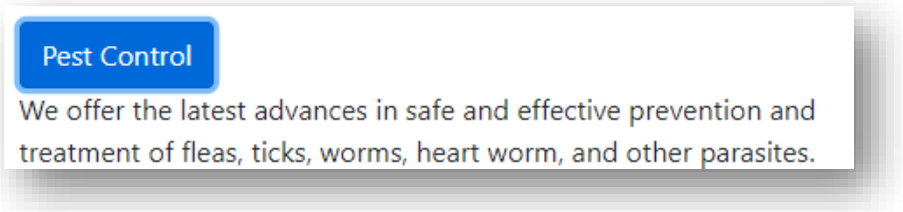
17. Collapse Accordions

Collapse and accordions are a set of related components that allow you to show or hide HTML content. Collapse is the simplest, and lets you show or hide content on a click. Accordions are similar but part of a bigger group. When one item shows, another hides, so accordions are essentially group of collapses. To set up a collapse, you create a link or a button that activates the collapse element. To that element, you add a **`data-toggle="collapse"`** and to target the element, you can use either an ID if it's an anchor tag or link, or you can use a data-target attribute if it's a button.

The element that's going to show or hide has the collapse class. Now as I mentioned, accordions are essentially a group of collapses put together. It does require an additional container, and one of the elements should be showing so you add the show class on that element, and you'll need to add the class of dropdown-menu, an it's common to put these in a card style.

Example)

```
<button class='btn btn-primary' data-toggle='collapse' data-
target='#petcontrol'>
  Pest Control</button>

<div id='petcontrol' class="collapse">
  We offer the latest advances in safe and effective prevention and treatment
of fleas, ticks, worms, heart worm, and other parasites.
</div>
```

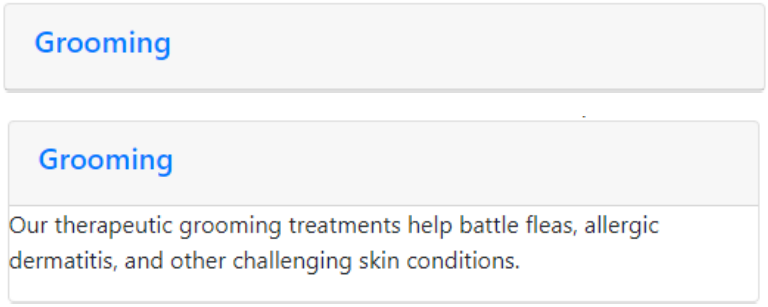


You can also you a collapse accordion by using cards

```
<!--create a container for all the cards-->
<div id='serviceAccordion' role='tablist'>
  <div class="card">
    <div class="card-header" role='tab' id='groomingheading'>
            <h5><a data-toggle='collapse' data-parent='serviceAccordion'
            href="#grooming">Grooming</a></h5>
    </div><!-- card header -->

    <div id="grooming" class='collapse' role='tabpanel'>
      <div class="card-block">
            <p>Our therapeutic grooming treatments help battle fleas,
            allergic dermatitis, and other challenging skin conditions.</p>
      </div>
    </div><!-- collapse -->
  </div><!-- card -->
```

If you want the accordion start showing the hidden content, you can add the word show after the class collapse

```
<div id="grooming" class='collapse show' role='tabpanel'>
```



Carousels

Carousels are the most popular Bootstrap component. It essentially allows you to create a slideshow that auto-advances and it gives you some controls to let you advance through the images. Now there's a ton of classes for the carousel so let's take a look at all of them. So the first thing you need to do is have a main component that has a class of carousel. That component needs to have at least a data-ride attribute with the word carousel. This will automatically activate the advancing of the carousel when the page loads. Now in addition to that, there will be another container that has a class of carousel-inner. This is what's going to have all your photos. Now each of the photos should have a div wrapping the photos with a class of carousel-item.

- `carousel`
- `data-ride="carousel"`
- `carousel-inner`

In addition to that, we have a few options. The first element that you want to show up when the carousel starts should have a class of active. That would go in the div, and you do want to crop and size the photos. Otherwise, the carousel will jump up and down.

- One element `active`
- Crop and size photos

In addition to that, we can add captions to a carousel with the carousel-caption class. So you would put another div, add this class to it, and that will go inside the div that has the image. So this is sort of what it looks like.

- `carousel-caption`

```
<div class="carousel-caption d-none d-md-block">
  <h3>...</h3>
  <p>...</p>
</div>
```

You can also make sure that you hide the carousel using the display classes if you want to.

So in addition to that, there's a number of navigation options so you can add these little arrows that will let you advance to the previous and next slide. The way you do that is by tying the carousel to the navigation with a **`data-target`** attribute. Whenever you use navigation, you need to make sure that you have that, and then you have two buttons that you can add. One with a class of **`carousel-control-prev`** and then you add an icon inside. And then also, you add a **`carousel-control-next`** with its icon as well.

You can also put indicators underneath each photo so that people can jump to any photo in the slide. And you do that by creating a section or another div with a class of carousel-indicators. Of course, that also gets a data-target so that Bootstrap knows which carousel you want to advance and then you add a slide number because, as you'll see, the indicators are sort of at the bottom. So this tells Bootstrap which photo to slide to.

- `data-target`
- `carousel-control-prev`
- `carousel-control-prev-icon`
- `carousel-control-next`
- `carousel-control-next-icon`
- `carousel-indicators`
- `data-target`
- `data-slide-to`

In order to set it up, you can do that with either a **`data- attribute`** and we talked about it earlier. It's called a data ride equals carousel. That will activate the carousel and make it start, or you can use JavaScript like this and pass it along some additional options.

- JavaScript or `data-`

```
$(function () {
    $('.carousel').carousel(OPTIONS)
})
```

Using data-attribute, the **interval** controls the amount of time in between each photo in milliseconds, **pause** will automatically pauses a carousel when somebody moves their mouse on top of that carousel. So by default, that will be on. You can turn it off by giving it a value of null if you want to.

- `interval` : 5000
- `pause` : hover | null
- `ride` : false
- `wrap` : true

Also, as I mentioned, the data **ride** attribute, you can set that to false if you want to. By default when you say data ride equals carousel, that will be on and that causes the carousel to automatically slide. If you don't want it to automatically advance, you can turn that false. Finally, there is an option to **wrap** the carousel which means that when it reaches the last photo, it'll go to the first photo.

Example)

```
<!-- carousel -->
<div class='carousel' data-ride='carousel'>
  <div class='carousel-item active'>
    <img class="img-fluid" src="images/carousel-lifestyle.jpg" alt="Lifestyle
Photo">
  </div>
  <div class='carousel-item'>
    <img class="img-fluid" src="images/carousel-fish.jpg" alt="Mission">
  </div>
  <div class='carousel-item'>
    <img class="img-fluid" src="images/carousel-vaccinations.jpg"
alt="Vaccinations">
  </div>
  <div>
    <h3>Pete, <small>owner of McAllister</small></h3>
    <p>"Wisdom Pet Medicine is the only clinic around that will even book pet
fish for appointments."</p>
  </div>
</div> <!--end of carousel-->
```

Adding caption to an image

```
<!-- carousel -->
<div class='carousel slide' data-ride='carousel'>
  <div class='carousel-item active'>
    <img class="img-fluid" src="images/carousel-lifestyle.jpg" alt="Lifestyle
Photo">
  <!-- adding a caption to the first image -->
    <div class='carousel-caption'>
      <h3>Pete, <small>owner of McAllister</small></h3>
      <p>"Wisdom Pet Medicine is the only clinic around that will even book
pet fish for appointments."</p>
    </div> <!--end of caption -->
  </div><!--end of first photo-->
  <div class='carousel-item'>
    <img class="img-fluid" src="images/carousel-fish.jpg" alt="Mission">
  </div>
  <div class='carousel-item'>
    <img class="img-fluid" src="images/carousel-vaccinations.jpg"
alt="Vaccinations">
  </div>
</div> <!--end of carousel-->
```

Next and previous buttons

```
<!-- next and prevous buttons -->
  <a class="carousel-control-prev" href='#carouselId' role='button' data-
slide='prev'>
    <span class='carousel-control-prev-icon'></span>
  </a>
  <a class="carousel-control-next" href='#carouselId' role='button' data-
slide='next'>
    <span class='carousel-control-next-icon'></span>
  </a> <!--end of next and prevous buttons-->
Page indicator
You can use an ordered list to create page indicators. This list should be
place within the carousel container
<!-- page indicator-->
  <ol class='carousel-indicators'>
    <li data-target='carouselId' data-slide-to='0'></li>
    <li data-target='carouselId' data-slide-to='1'></li>
    <li data-target='carouselId' data-slide-to='2'></li>
  </ol>
```

Scrollspy

Scrollspy is a really cool component that allows you to keep track of the scroll of the page and modifies classes according to the position of your elements. So the way that you use it is by creating a data attribute called scroll, and you do that on the container of the element that you want to track the scrolling in. Now that element also needs to have a position of relative, and you also need to target that element with a data target attribute, and then an ID. Data target is usually how we put together what we are doing with the object that we're targeting.

- `data-spy="scroll"`
- `position: relative`
- `data-target="ID"`
- `fixed-top`
- `data-offset`

Example)

```
<body data-spy='scroll' data-target='#navbar-site' data-offset='80'>
<nav id="navbar-site" class="fixed-bottom navbar navbar-dark bg-dark navbar-
expand-sm">
  <div class="container">
    <ul class="navbar-nav">
      <li class="nav-item"><a class="nav-link"
href="#mission">Mission</a></li>
      <li class="nav-item"><a class="nav-link"
href="#services">Services</a></li>
      <li class="nav-item"><a class="nav-link" href="#staff">Staff</a></li>
      <li class="nav-item"><a class="nav-link"
href="#testimonials">Testimonials</a></li>
    </div><!-- navbar-nav -->
  </ul><!-- container -->
</nav>
```

Modals

Modals allow you to show and hide content that is in an element showing up as an overlay on top of your page. Like with other components, modals have two parts: the trigger, which can be a button or a link, as well as a piece of content that you want to show in the modal. If you use a link, you'll need to target an ID that you want to use as your modal. If you're using a button, then you use the data-target attribute. The link, or the button, will need a data-toggle="modal" attribute. Now the content you want to show as the modal needs to have the modal class. So this is our main container for the entire element.

- Button or Link
- `#id` or `data-target`
- `data-toggle="modal"`
- `modal`

In addition to the modal class, there's some other classes that set up the structure of how your modal works. Modal-dialog is an extra container that is used for some additional spacing. There's another container class called modal-content that you can use for the main content of the modal.

You can also use an optional header for a title in your modal, and you should probably use a header in all your modals.

Now the modal-body is where you put your main content, and there is an optional footer for things like close buttons. There's a couple of other options that you can use if you are using a headline, you can also use modal-title to handle the headline that you're using for that title. You should have at least one element with at a data-dismiss attribute of modal, and that will get rid of the modal and take you back to your page.

- `modal-dialog`
- `modal-content`
- `modal-header`
- `modal-body`
- `modal-footer`

Modal Options

- `modal-title`
- `data-dismiss="modal"`

Reference

Window size, margin, and padding

Bootstrap has a wide range of responsive margin and padding utility classes. They work for all breakpoints:

xs (<=576px), **sm** (>=576px), **md** (>=768px), **lg** (>=992px) or **xl** (>=1200px))

The classes are used in the format:

{property}{sides}-{size} for xs & **{property}{sides}-{breakpoint}-{size}** for sm, md, lg, and xl.

m - sets margin

p - sets padding

---

t - sets margin-top or padding-top

b - sets margin-bottom or padding-bottom

l - sets margin-left or padding-left

r - sets margin-right or padding-right

x - sets both padding-left and padding-right or margin-left and margin-right

y - sets both padding-top and padding-bottom or margin-top and margin-bottom

blank - sets a margin or padding on all 4 sides of the element

---

0 - sets **margin** or **padding** to 0

1 - sets **margin** or **padding** to .25rem (4px if font-size is 16px)

2 - sets **margin** or **padding** to .5rem (8px if font-size is 16px)

3 - sets **margin** or **padding** to 1rem (16px if font-size is 16px)

4 - sets **margin** or **padding** to 1.5rem (24px if font-size is 16px)

5 - sets **margin** or **padding** to 3rem (48px if font-size is 16px)

auto - sets margin to auto

Bibliography

Duckett, J. (2016). *HTML and CSS Design and build websites.* Indianapolis: John Wiley and Sons Inc.

Duckett, J. (2016). *JavaScript and JQuery: interactive front-end developer.* Indianapolis: John Wiley and Sons Inc.

HTML, CSS, JavaScript, JQuery, and Bootstrap. (2017, June). Retrieved from w3schools: www.w3schools.com

Some material compiled from www.lynda.com (May 2018), www.w3schools.com (2020), www.coursera.org (2020)