Introduction to Bootstrap: Objectives and Outcomes

In this lesson, you will be given a quick overview of front-end UI frameworks, and an introduction to Bootstrap. The exercises will introduce you to getting started with Bootstrap for your web project. At the end of this lesson, you will be able to:

- Identify the purpose of using front-end UI frameworks in web design and development
- Set up a project with Bootstrap support
- Configure a web project to use Bootstrap
- Become familiar with the basic features of Bootstrap

Introduction to Bootstrap

Bootstrap is a UI framework. A UI framework, in general, is a *collection of templates (html, css, js) for UI components*. Possible components include forms, buttons, tables, nav bars, dropdowns, alerts, tabs, modals, accordions, carousels, and typography.

Frameworks make it easy to do *responsive web design*, which is the ability of a website to adapt to many devices/ screen sizes/ resolutions/ connection speeds/ etc. Frameworks also handle cross-browser compatibility, and speed production by abstracting many low-level details.

At the present time (2016), the trend is to design for mobile first, then adapt to larger screen sizes.

Bootstrap has/does everything in the previous three paragraphs.

Getting Started with Bootstrap

There are 3 ways to use boostrap

1. Download & install the minified files from http://getbootstrap.com
2. Obtain the source in Less or Sass preprocessor format
3. Link to the Bootstrap CDN (content distribution network) in your document
4. Sources for all: http://getbootstrap.com/getting-started/#download

When using method 1, the directory tree will look like this (Table 1)

bootstrap/			
	css/	js/	fonts/
	– – bootstrap.css	– – bootstrap.min.js	– – glyphicons-halflings-regular.eot
	– – bootstrap.min.css	– – bootstrap.min.js	– – glyphicons-halflings-regular.svg
	– – bootstrap-theme.css		– – glyphicons-halflings-regular.ttf
	– – bootstrap-theme.min.css		– – glyphicons-halflings-regular.woff

Table 1

Once the Bootstrap .zip has been extracted, reference the files by including these lines in the html <head>:

and these lines at the end of the html <body> (href and src attributes assume a full path to the bootstrap directory

Bootstrap uses a global, fixed width `<div class="container"></div>`, just inside the <body>, to define screen widths for different devices. Full width containers are available with the *container-fluid* class. Other bootstrap classes include *row*, *column*, *jumbotron*

Bootstrap lays out items in a grid of container divs. In order for the grid system to work, divs with the

Code 1:

```
<meta charset="utf-8">
<meta http-equiv="X-UA-Compatible" content="IE=edge">
<meta name="viewport"  content="width=device-width, initial-scale=1">

<link href="css/bootstrap.min.css" rel="stylesheet">
<link href="css/bootstrap-theme.min.css" rel="stylesheet">
```

appropriate classes must be used

- `<div class="container"></div>`

Code 2:

```
<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.3/jquery.min.js">
</script>

<!--
  Include all compiled plugins (below), or include individual files as needed
-->
<script src="js/bootstrap.min.js"></script>
```

  - directly inside the <body> tag
  - fixed width
- `<div class="row"></div>`
  - groups information into horizontal collections
  - automatically broken into 12 columns
  - rows must always be inside containers
- `<div class="col-sm-*"></div>`
  - spans * number of the 12 default columns
  - 4 subclasses: xs, sm, md, lg; further specify behavior by screen size
  - columns must always be inside rows
- `<div class="jumbotron"></div>`
  - used for showcasing content
  - creates a gray-background banner inside the global fixed-width container div
  - place jumbotron outside the container to span entire page

Setting Up a Development Environment

Consider using Sublime Text or Brackets as an editor. Netbeans also has an HTML5/JavaScript bundle.

1. Create Project Directory
   - Create a folder called `conFusion`. This will be the parent directory for the *conFusion* website.
2. Download the Bootstrap Files
   - Download & copy the Bootstrap .zip into the Project Directory `conFusion/` from http://getbootstrap.com/getting-started/#download
3. Move Bootstrap Folders into Project Folder
   - Move `css/`, `fonts/`, and `js/` into from `bootstrap-x.x.x-dist/` into `conFusion/`.
4. Download the project HTML template
   - Download into `conFusion/` from https://d396qusza40orc.cloudfront.net/phoenixassets/web-frameworks/index.html

Exercise 1 – Getting Started with Bootstrap

Apply basic styling with Bootstrap classes:
`conFusion/Exercise_Instructions_Getting_Started_with_Bootstrap.html.maff`

In the HTML File **Ristorante Con Fusion.html**

1. Add the Required Tags
   i. Add meta tags and css links to head (see example)
   ii. Add javascript links at end of body
2. Add the container class
   i. `<div class=“container”></div>` </header> and <footer>
   ii. (this creates *de facto* margins around the main content)
3. Group content into 3 rows
   i. `<div class=“row”></div>` around the 3 major content groups
4. Prepare the header
   i. `<header class=“jumbotron”>`
   ii. `<div class=“container”></div>` to first div inside <header>
   iii. `<div class=“row”></div>` to first div inside container
5. Prepare the footer
   i. `<div class=“container”></div>` to first div inside <footer>
   ii. `<div class=“row”></div>` to first div inside container

Additional Resources

Responsive Design and Bootstrap Grid System: Objectives and Outcomes

In this lesson, you will be given an overview of responsive web design and an introduction to the

Bootstrap grid system. The exercises will concentrate on enhancing your web project using the Bootstrap grid in order to make it responsive. At the end of this lesson, you will be able to:

- Understand the reasons for using responsive web design in a web project
- Use the Bootstrap grid system to design responsive websites
- Add your own custom CSS classes to a Bootstrap based web project

Responsive Design

A responsive website automatically adapts its layout to many different screen sizes. It does this via the *viewport* property of a device. Functionally this means that the page will move/resize/show/hide elements based on viewing screen.

Responsive design requires 3 capabilities from a UI framework:

1. A grid system
2. Fluid image support
3. Media query support

Media Queries

A media query is a css query using the `@media` attribute. Using this attribute, a browser can query a device's *viewport* property to find the device's screen size (in pixels??). Then, it can render the page based on css style rules pertaining to that specific screen size, using a syntax like this

Code 3, CSS:

```
@media screen and (min-width:600px) {
      /* CSS styles customized for this (desktop) screen size*/
      .container{width: 585px}  /*for example*/
}
```

The Bootstrap Grid System

The Bootstrap grid system is designed around, in order of precedence, *containers* > *rows* > *columns*. This organization allows web pages to responsively change their rendering to different screen sizes. The actual rendering rules are executed according to a device's *viewport* property; we give our page the ability to query the viewport via this meta tag in <head>:

- `<meta name="viewport" content="width=device-width, initial-scale=1">`
  - enables querying of a device's *viewport* property
  - specifies that screen width is equal to the device width
  - ensures that web content is scaled to 100% of this width
  - source link

The Container Class

Container class divs set the available width for the page content. Specifically, the bootstrap 'container' class sets this width to the device screen width.

The Row Class

Row class divs are automatically divided into 12 columns. Use any of the column subclasses[1] to specify how many columns a column div should span. All column spans in a row should add up to 12, or a multiple of 12. There is always 30 px of *gutter width* (whitespace) between adjacent pieces of columnar content.

[1] See next section: The Column Class and Its Subclasses

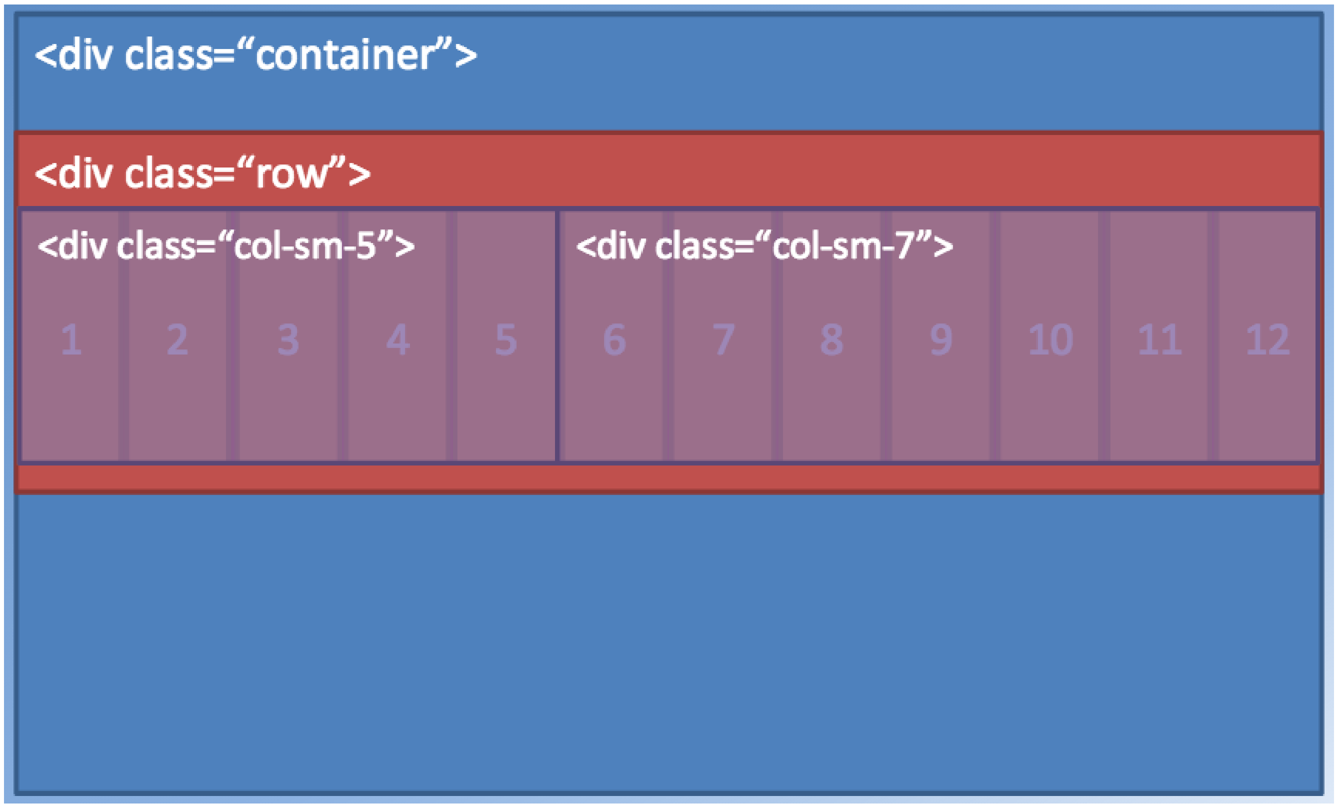


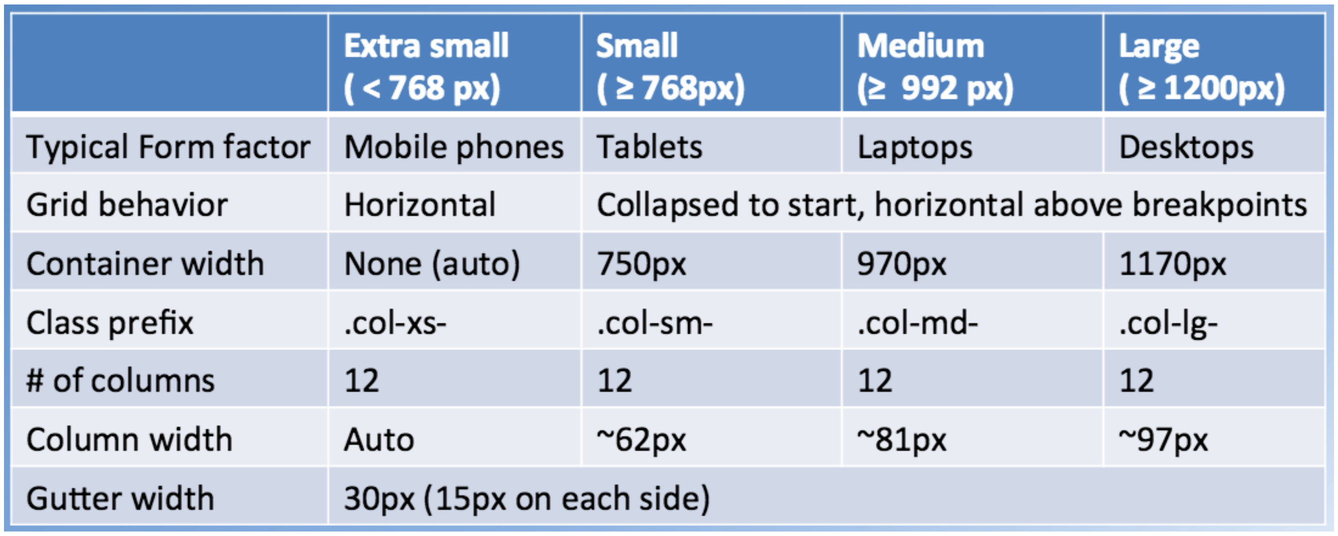
Illustration 1: The Bootsrap row class has 12 columns by default.

The Column Class and Its Subclasses

Column class divs have 4 subclasses for different screen sizes. A single column has a different base-width in each size. The column subclasses are specified with the following syntax ('*' specifies the number of columns to span)

- xs – (< 768 px) for mobile phone screens
  - `<div class="col-xs-*"></div>`
- sm – (≧ 768 px) for tablets
  - `<div class="col-sm-*"></div>`
- md – (≥ 992 px) for desktop monitors
  - `<div class="col-md-*"></div>`

- lg – (≥ 1200 px) for very large monitors
  - `<div class="col-lg-*"></div>`



	Extra small (< 768 px)	Small (≥ 768px)	Medium (≥ 992 px)	Large (≥ 1200px)
Typical Form factor	Mobile phones	Tablets	Laptops	Desktops
Grid behavior	Horizontal	Collapsed to start, horizontal above breakpoints		
Container width	None (auto)	750px	970px	1170px
Class prefix	.col-xs-	.col-sm-	.col-md-	.col-lg-
# of columns	12	12	12	12
Column width	Auto	~62px	~81px	~97px
Gutter width	30px (15px on each side)			

llustration 2: Behaviors of the 4 column subclasses

'xs' columns are always stacked vertically. Larger subclasses will arrange horizontally, and stack when viewed on smaller devices. An example is shown below, in which two 'xs/sm' columns are laid out on xs, sm, md, and lg-sized screens.

Behavior of Column Subclasses

Extra Small Screens

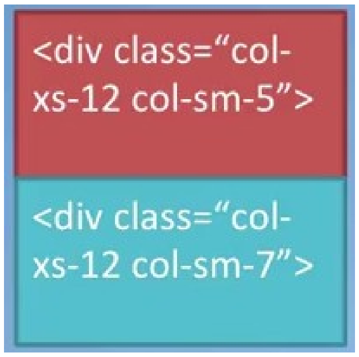

<div class="col-xs-12 col-sm-5">
<div class="col-xs-12 col-sm-7">

Illustration 3: xs screens

Small, Medium, and Large Screens

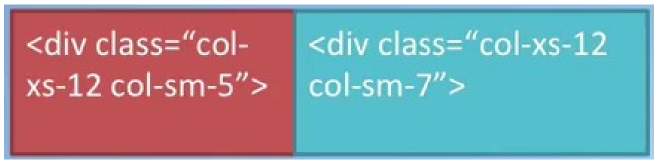

<div class="col-xs-12 col-sm-5">
<div class="col-xs-12 col-sm-7">

Illustration 4: sm, md, and lg screens

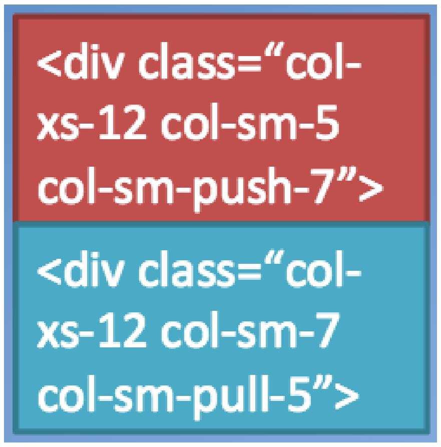
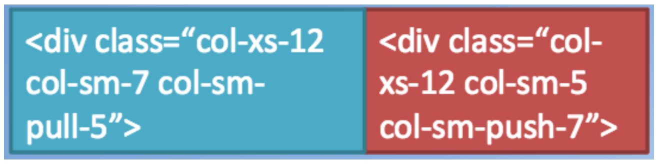
On large screens, the layout defaults to the largest (smaller) size specified; in this case, *sm*, with the divs spanning 5 and 7 sm, md, or lg-sized columns, respectively. On small screens, the layout defaults to the current screen size, *xs*, with all divs spanning all 12 of the tiny, xs-sized columns.

Pull and Push Subclasses

A problem arises with Bootstrap's column-stacking behavior. By default, depending on screen size, the first div in a row will be on top, or leftmost in a row. What if you want the columns to display in a different order based on screen size? Pull and push classes solve this problem by overriding the default layout rules.

In the previous example, the red div displayed first (top and left) in all screens. In the following example, we want the red `<div class="col-xs-12 col-sm-5 col-sm-push-7">` to appear first (top) on xs screens, and second (right) sm or larger screens.

Behavior of Push / Pull Subclasses

Extra Small Screens	Small, Medium, and Large Screens
 Illustration 5: xs with push/pull	 llustration 6: sm, md, lg with push/pull

Adding the *push* and *pull* subclasses to the *sm* subclass does two things when rendered on large screens:

1. `<div class="col-xs-12 col-sm-5 col-sm-push-7">`
   - pushes the red div 7 columns to the right
2. `<div class="col-xs-12 col-sm-7 col-sm-pull-5">`
   - pulls the blue div 5 columns to the left

This effectively swaps the divs when laid out horizontally. In this example, the amount of push or pull is the 12-complement of the div width. The ratio is more complex with three or more divs in a row, but it must work out, or the divs will overlap each other.

The Offset Subclasses

Push and pull will always force column-divs to be adjacent. If you want whitespace in between them, use *offset*: `<div class="col-md-4 col-md-offset-*">`. The offset subclass will force * empty columns (whitespace) to precede the div. The offset is measured from the first preceding element, which means you can do <space> <div>, or <div> <space> <div>.

Nesting a Column

Nest columns by creating a row within a column, then additional columns within that row. This works because rows and columns aren't really hierarchical. Rows simply create groups of content to be laid out horizontally, and columns simply declare how much horizontal space each piece of content should occupy. This is demonstrated in the following example.

Code 3, HTML: Nesting columns within columns

```
<div class="container">
  <div class="row">
    <div class="col-sm-5"></div>
    <div class="col-sm-7">
      <div class="row"> <!-- this row is divided into a full 12 columns -->
        <div class="col-sm-9"></div>
        <div class="col-sm-3"></div>
      </div>
    </div>
  </div>
</div>
```

Notice that within each row, column spans add to 12.

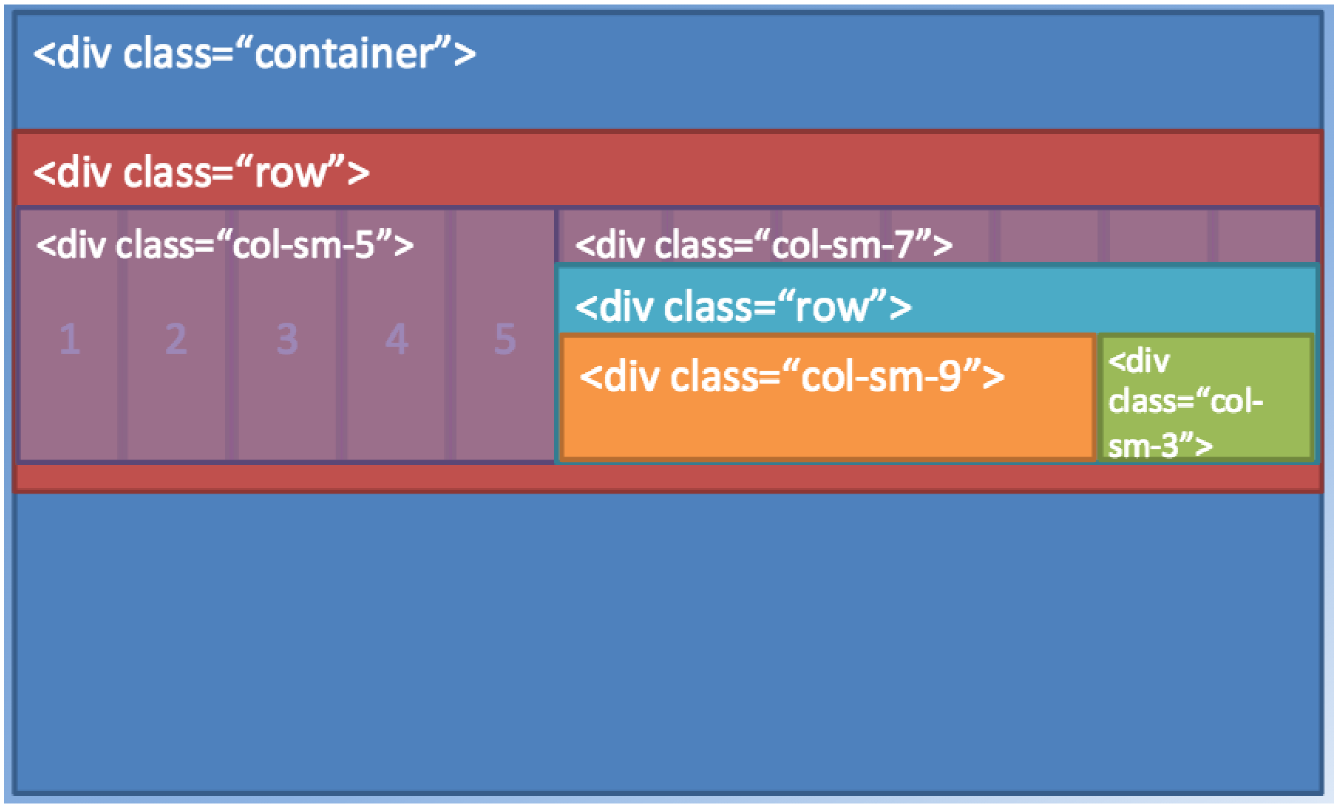


Illustration 7: Nesting columns

Summary of Bootstrap Grid Classes

- container – sets page width to device screen width
- row – creates horizontal groups

- col – spans any number of the default 12 columns per row
- push – pushes divs to the right
- pull – pulls divs to the left
- offset – inserts whitespace to the left of the div

Exercise 2 – Responsive Design & Grid Systems

Add Bootstrap classes and css style rules to the html created in Exercise 1. Make sure you have the meta tag … inside <head>.

Applying Column Classes to Rows

1. Sadf
2. asdf
3. asd

Using *Push, Pull,* and *Offset*

1. Asdf
2. asdf
3. asdf

Styling Lists

Using Custom CSS Classes

Quiz 2 – Responsive Design & Grid Systems

Additional Resources

Navigation and Navigation Bar: Objectives and Outcomes

In this lesson, you will be given an overview of navigation design and the importance of providing appropriate navigation support within your website. You will learn about support for navigation design elements available in Bootstrap, including the Navbar and Breadcrumbs. Other navigation aids will be covered in subsequent modules. In addition, the use of icon fonts in web page design will be covered. The exercises will concentrate on adding a responsive navigation bar to the website. At the end of this lesson, you will be able to:

- Understand the need for navigation support in a web project
- Use the Bootstrap navigation features including the Navbar and breadcrumbs in providing navigation support in websites
- Use icon fonts for decorating your website with meaningful graphical elements

Navigation and Navigation Bars

The Bootstrap NavBar

```
Only the <button> element may be used in Bootstrap navbars. HTML anchors won't work.
```

Icon Fonts

Exercise 3 – Navbar

Quiz 3 – Navigation Bars

Additional Resources

```
<meta charset="utf-8">
<meta h@p-equiv="X-UA-Compatible" content="IE=edge">
<meta name="viewport"  content="width=device-width, initial-scale=1">

<link href="css/bootstrap.min.css" rel="stylesheet">
<link href="css/bootstrap-theme.min.css" rel="stylesheet">

<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.3/jquery.min.js">
</script>

<!--
```

```
  Include all compiled plugins (below), or include individual files as needed
-->
<script src="js/bootstrap.min.js"></script>
```

http://www.zeninvader.com/css/bootstrap-3-grid-system-explained

Buttons & Forms

Objectives and Outcomes

In this lesson we review the support for user input through the use of buttons and forms in a web page. We review Bootstrap button classes and Forms classes. At the end of this lesson you will be able to:

- Create and style buttons on a web page using Bootstrap button classes
- Create and style forms on a web page using Bootstrap form classes

User Input

The entire body of current user input can be accomplished with 3 major tags

1. `<a>` anchor tags provide hyperlinks
2. `<button>` tags create action buttons
3. `<form>` tags create input forms
    - `<input>` tags create the various kinds of input elements – text box, slider, etc

Bootstrap Buttons

Buttons have 1 mode of interaction; it's clicked or not clicked.

Button behavior depends on its location. A button inside a form will usually only submit the form info. Buttons outside a form have a more general scope. Some buttons are actually styled anchor tags. Normal button rules don't apply to them.

The bootstrap button class (`class="btn"`) may be applied to 3 html elements: `<a>`, `<button>`, `<input>`. The buttom class makes any of these elements look like a button.

Buttons may be grouped into toolbars (`btn-toolbar`) and groups (`btn-group`, `btn-group-vertical`), with the hierarchy toolbar > group. Button toolbars may have multiple groups, and buttons in a group are laid out contiguously. An advantage of this is styling a collection of buttons all at once, by chaining the appropriate style modifier to the toolbar or group class. Examples follow:

Code:

```
<div class="btn-toolbar" role="toolbar">

<div class="btn-group btn-group-sm" role="group">
```

Button Options: The Bootstrap Color Palette

Bootstrap designates six contextual classes that are correlated with six different colors.

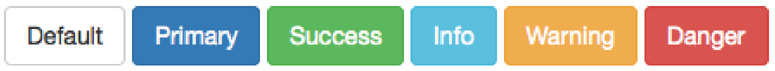


Illustration 1: The Bootstrap Content Color Classes

These colors may be applied by chaining the color class name to an element class name with a hyphen:

```
<!-- Standard button -->
<button type="button" class="btn btn-default">Default</button>

<!-- Provides extra visual weight and identifies the primary action in a set of buttons
-->
<button type="button" class="btn btn-primary">Primary</button>

<!-- Indicates a successful or positive action -->
<button type="button" class="btn btn-success">Success</button>

<!-- Contextual button for informational alert messages -->
<button type="button" class="btn btn-info">Info</button>

<!-- Indicates caution should be taken with this action -->
<button type="button" class="btn btn-warning">Warning</button>

<!-- Indicates a dangerous or potentially negative action -->
<button type="button" class="btn btn-danger">Danger</button>

<!-- Deemphasize a button by making it look like a link while maintaining button behavior
-->
<button type="button" class="btn btn-link">Link</button>
```

You can also specify size by hyphen-chaining `xs`, `sm`, or `lg`. The default size would be `btn-md`, although `btn-md` doesn't exist in this context. The `btn-block` subclass makes buttons span their entire column-div. Multiple block buttons in one column will stack.

Button Class Summary

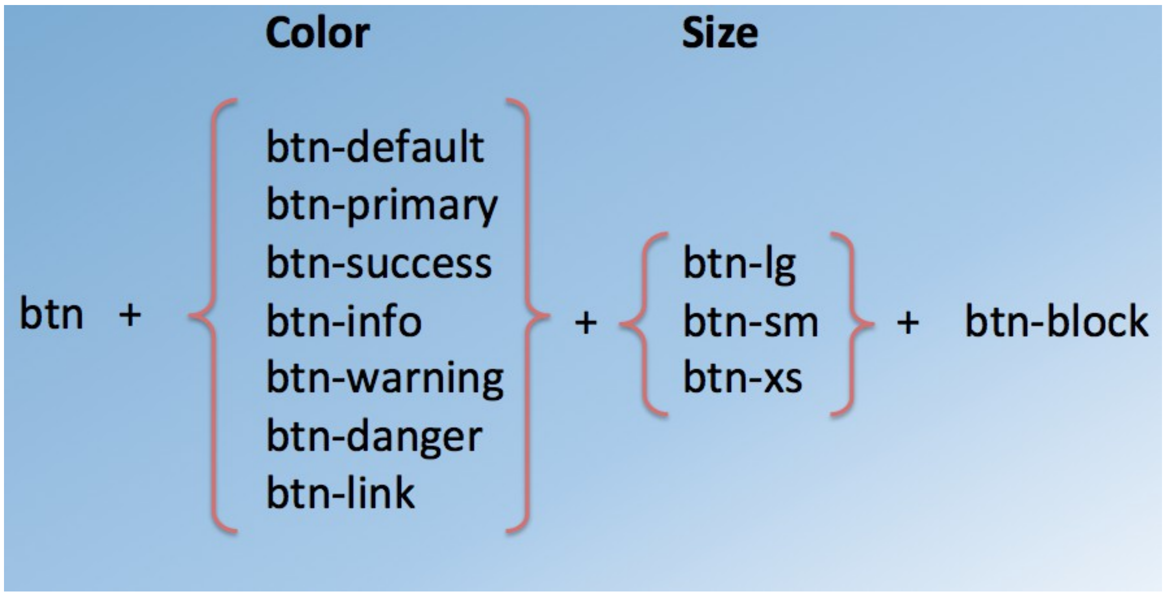


Illustration 2:

Bootstrap Forms

Exercise 1 – Buttons and Forms

User Input: Additional Resources

Displaying Content: Tables, Panels, Wells

Objectives and Outcomes

In this lesson we will be reviewing the support for tables in Bootstrap. In addition two components in Bootstrap, viz. Panels and Wells and their application to display content will be presented. At the end of this lesson you will be able to:

- Present and style tabular data in a table form using Bootstrap support for tables
- Display content using panels and wells on a web page using Bootstrap panel and well components

Bootstrap Tables

Bootstrap Panels and Wells

Exercise 2 – Tables, Panels, and Wells

Displaying Content: Additional Resources

Images and Media: Images, Thumbnails, Media Objects

In this lesson we will look at the use of images and media on websites. In particular we will review the Bootstrap classes to support the inclusion of images and media, supporting responsiveness of images and media, and the use of these as thumbnails and part of other components, in particular the media component. At the end of this lesson you will be able to:

- Use images and media and include them in your website
- Support responsive images and media using responsive Bootstrap classes for images and media
- Use thumbnails and media components using Bootstrap classes

Objectives and Outcomes

Bootstrap Images and Media

Exercise 3 – Images and Media

Images and Media: Additional Resources

Alerting Users: Labels, Badges, Alerts, Progress Bar

Objectives and Outcomes

In this lesson we examine various ways of delivering alert information to users. We examine labels, badges, alerts and progress bars. At the end of this lesson, you will be able to:

- Include labels and badges in your web page
- Create, style and include alerts in your web page
- Appreciate the use of progress bars and controlling the state of the progress bars.

Alerting Users

Labels

Badges

Alerts

Progress Bars

Exercise 4 – Alerting Users

Alerting Users: Additional Resources