**CSS Frameworks**

Given that writing CSS to custom-style your Web page is cumbersome and difficult, there have been attempts to develop CSS frameworks to make the developer’s job easier. Using such frameworks not only makes the job of the developer easy, but they also enable the developer to preserve the same look and feel for different pages of an application making the application have a consistent view across different pages, which is important.

This week we will look into 4 very popular CSS frameworks: Bootstrap, Materialize CSS, Bulma CSS, Tailwind CSS.

**Bootstrap [**<https://getbootstrap.com/>, <https://www.w3schools.com/bootstrap4/default.asp>**]**

Bootstrap is the most popular HTML, CSS, and JavaScript framework for developing responsive, mobile-first websites. Bootstrap 4 is the new version that supersedes version 3 with new components, faster stylesheet, and more responsiveness. It uses flexbox for layout management as opposed to floats, which is what Bootstrap 3 uses.

Here are the advantages of Bootstrap:

* **Easy to use**: Anybody with just basic knowledge of HTML and CSS can start using Bootstrap
* **Rapid development**: Using Bootstrap components you can quickly develop a nice, responsive web page. This is in fact why you would use any framework.
* **Responsive features**: Bootstrap's responsive CSS adjusts to phones, tablets, and desktops
* **Mobile-first approach**: In Bootstrap, mobile-first styles are part of the core framework
* **Browser compatibility**: Bootstrap 4 is compatible with all modern browsers (Chrome, Firefox, Internet Explorer 10+, Edge, Safari, and Opera)

Here is what you need to include into your Web pages to make use of Bootstrap 4 components.

|  |
| --- |
| <head>  <title>Simple Page with Bootstrap 4</title>  <!-- These two are for responsive pages -->  <meta charset="utf-8">  <meta name="viewport" content="width=device-width, initial-scale=1">  <!-- Latest compiled and minified CSS -->  <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.4.1/css/bootstrap.min.css">  <!-- The following Javascript libraries are necessary only if you are using Bootstrap components  that make use of Javascript such as modals, tooltips, popovers etc.  However, if you just use the CSS part of Bootstrap, you don't need them. -->  <!-- jQuery library JavaScript-->  <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js"></script>  <!-- Popper JS -->  <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.16.0/umd/popper.min.js"></script>  <!-- Latest compiled JavaScript -->  <script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.4.1/js/bootstrap.min.js"></script>  </head> |

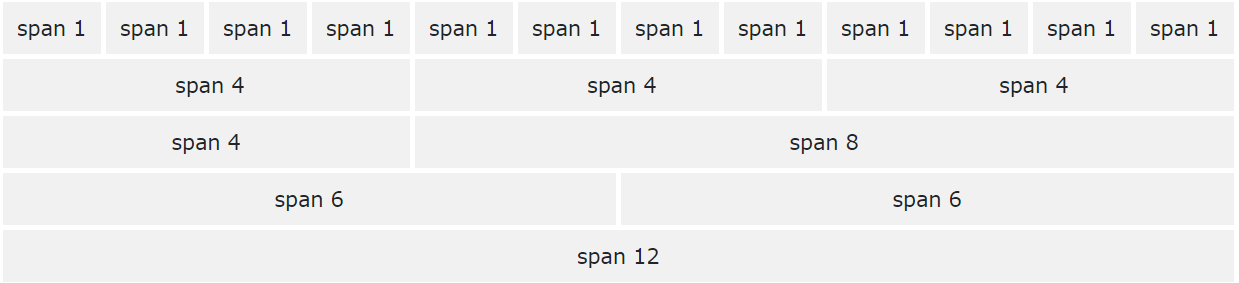
Look at 04-CSS-Frameworks/01-FirstPage/index1.html

As you can see from the above example, Bootstrap is nothing but a CSS library. So, before using the CSS components defined in Bootstrap, you need to include the style sheet in you Web page using the <link> element. You can either download Bootstrap 4 and link the downloaded .css file to your web page, or you can point to a version stored in a content distribution network (CNN) as we did here. “bootstrap.min.css” is so-called the minimized version of Bootstrap with all comments and unnecessary space/newline characters removed so as to minimize the size of the CSS file. When you download the entire Bootstrap style file, you will also get the long, commented version, which you can have a look at to understand how it is implemented.

After the Bootstrap CSS file, you also need to include 3 JavaScript (JS) libraries as listed above in this order: First jQuery, then Popper, and then Bootstrap JS plugins. As we stated in the comments, these JS libraries are necessary only if you are using Bootstrap components that make use of JS such as modals, tooltips, popovers etc. However, if you just use the CSS part of Bootstrap, you don't need them. Although we have included these JS files in the <head> section, it is usually a good practice to include them at the end of the body section of your Web page as we will discuss next week when we start covering JS.

**Bootstrap 4 Grid System [**<https://www.w3schools.com/bootstrap4/bootstrap_grid_system.asp>**]**

Bootstrap's grid system is built with flexbox and allows up to 12 columns across the page as shown below:



The grid system is responsive, and the columns will re-arrange automatically depending on the screen size. Make sure that the sum adds up to 12 or fewer (it is not required that you use all 12 available columns).

The Bootstrap 4 grid system has five classes:

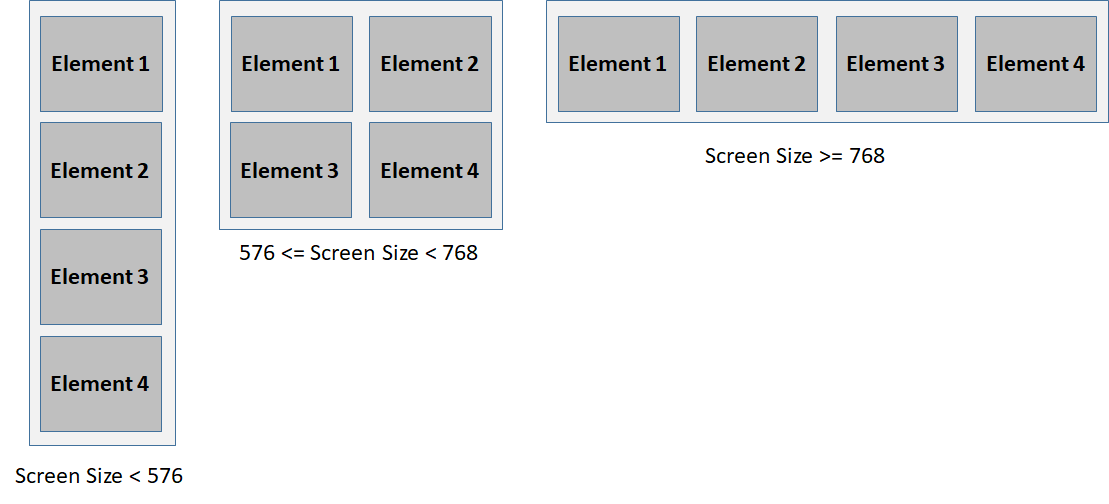
* **.col-** (extra small devices - screen width **less than** **576px**)
* **.col-sm-** (small devices - screen width **equal to or greater than 576px**)
* **.col-md-** (medium devices - screen width **equal to or greater than 768px**)
* **.col-lg-** (large devices - screen width **equal to or greater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

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Extra small** | **Small** | **Medium** | **Large** | **Extra Large** |
| **Class prefix** | .col- | .col-sm- | .col-md- | .col-lg- | .col-xl- |
| **Screen width** | <576px | >=576px | >=768px | >=992px | >=1200px |

The classes above can be combined to create more dynamic and flexible layouts. Note that each class scales up, so if you wish to set the same widths for *sm* and *md*, you only need to specify *sm*.

To make use of Bootstrap’s grid, create a row (<div class="row">). Then, add the desired number of columns (tags with appropriate .col-\*-\* classes) for each element on the row for each device. Default is 100% for mobile devices. You then specify how many columns an element should occupy as the screen size gets bigger.

As an example, assume that we have 4 elements on a row. We want each element to occupy the entire row (100%) for mobile devices (screen size <= 576). For screen sizes up to 768 (sm) we want the first two elements to appear size by size (50% of the row) and the last two elements to appear side by side (50% each) on a second row below the first two elements. Finally, for screen sizes >=768, we want all 4 elements to appear side-by-side each occupying 25% of the row. Here is a picture of how we want 4 elements to appear depending on the screen size:



Here is how we can achieve this using Bootstrap: Define a row, and put the elements inside it. For each element, specify how many columns the elements should occupy for each screen size. Recall that Bootstrap is a mobile-first design. So if you do not specify any width for the mobile screen size of (<576 px), then it is assumed that the element occupies the entire row, i.e., its width is 100%.

|  |
| --- |
| <div class="row">  <div class="**col-sm-6 col-md-3**">  <h3>Column 1</h3>  <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit...</p>  <p>Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris...</p>  </div>  <div class="**col-sm-6 col-md-3**">  <h3>Column 2</h3>  <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit...</p>  <p>Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris...</p>  </div>  <div class="**col-sm-6 col-md-3**">  <h3>Column 3</h3>  <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit...</p>  <p>Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris...</p>  </div>  <div class="**col-sm-6 col-md-3**">  <h3>Column 4</h3>  <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit...</p>  <p>Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris...</p>  </div>  </div> |

Look at 04-CSS-Frameworks/02-GridSystem/index1.html

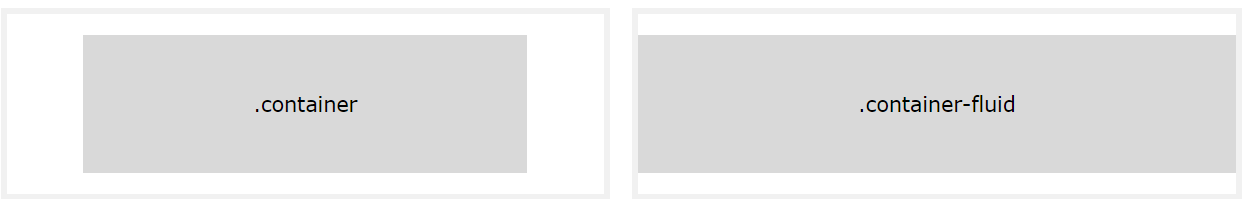
Not all columns has to have the same width as is the case for the above example. You can have columns of different sizes. Recall our responsive Web page from last week that we designed using flexbox. Here is that same responsive Web page with Bootstrap:

Look at 04-CSS-Frameworks/03-ExampleResponsiveWebPage/index1.html

**Bootstrap Containers**

It is customary to put HTML elements inside a Bootstrap container. Containers are used to add padding, margin, border, text color etc. for the content stored inside. There are two container classes available:

* The **.container** class provides a responsive **fixed width container**
* The **.container-fluid** class provides a full width container, **spanning the entire width of the viewport**



**max-width** will change on different screen sizes for the **.container** class as follows:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | *Extra small*  *<576px* | *Small*  *≥576px* | *Medium*  *≥768px* | *Large*  *≥992px* | *Extra large*  *≥1200px* |
| **max-width** | 100% | 540px | 720px | 960px | 1140px |

**.container-fluid** class however will always span the entire width of the screen (width is always 100%).

Look at 04-CSS-Frameworks/01-FirstPage/index2.html

By default, containers have 15px left and right padding, with no top or bottom padding. Therefore, we often use spacing utilities, such as extra padding and margins to make them look even better. For example, “**pt-3**” class adds a top padding of 16px":

|  |
| --- |
| <div class="container pt-3"></div> |

Other utilities, such as *borders* and *colors*, are also often used together with containers.

You can also use the .container-sm|md|lg|xl classes to create responsive containers. The max-width of the container will change on different screen sizes/viewports as follows:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | *Extra small*  *<576px* | *Small*  *≥576px* | *Medium*  *≥768px* | *Large*  *≥992px* | *Extra large*  *≥1200px* |
| **.container-sm** | 100% | 540px | 720px | 960px | 1140px |
| **.container-md** | 100% | 100% | 720px | 960px | 1140px |
| **.container-lg** | 100% | 100% | 100% | 960px | 1140px |
| **.container-xl** | 100% | 100% | 100% | 100% | 1140px |

Look at 04-CSS-Frameworks/01-FirstPage/index3.html

Look at 04-CSS-Frameworks/02-GridSystem/index2.html

Look at 04-CSS-Frameworks/03-ExampleResponsiveWebPage/index2.html

**Typography [**<https://www.w3schools.com/bootstrap4/bootstrap_typography.asp>**]**

Bootstrap 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. Furthermore, all <p> elements have margin-top: 0 and margin-bottom: 1rem (16px by default).

Here are some of the typographic classes provided by Bootstrap. For the complete list, refer to the link above.

|  |  |
| --- | --- |
| .page-header | A page header with a border |
| .small | Indicates smaller text (set to 80% of the size of the parent) |
| .lead | Makes a paragraph stand out |
| .font-weight-bold | Bold text |
| .font-italic | Italic text |
| .text-center | Makes the text center-aligned (.text-left, .text-right) |
| .text-lowercase | Makes all the text lowercased |
| .text-uppercase | Makes all the text uppercased |

Look at 04-CSS-Frameworks/04-Typography/index.html

**Colors [**<https://www.w3schools.com/bootstrap4/bootstrap_colors.asp>**]**

Bootstrap has some contextual classes that can be used to provide "meaning through colors". The classes for text colors are: .**text-muted**, .**text-primary**, .**text-success**, .**text-info**, .**text-warning**, .**text-danger**, .**text-secondary**, .**text-white**, .**text-dark**, .**text-body** (default body color/often black) and .**text-light**:

Look at 04-CSS-Frameworks/05-Colors/index1.html

The classes for background colors are: .**bg-primary**, .**bg-success**, .**bg-info**, .**bg-warning**, .**bg-danger**, .**bg-secondary**, .**bg-dark** and .**bg-light**. Note that background colors do not set the text color, so in some cases you'll want to set the text-color with a .text-\* class.

Look at 04-CSS-Frameworks/05-Colors/index2.html

**Jumbotron [**<https://www.w3schools.com/bootstrap4/bootstrap_jumbotron.asp>**]**

A jumbotron indicates a big grey box for drawing extra attention to some special content or information. Inside a jumbotron you can put nearly any valid HTML, including other Bootstrap components. Use a <div> element with class .**jumbotron** to create a jumbotron:

Look at 04-CSS-Frameworks/06-Jumbotron/index1.html

If you want a full-width **jumbotron** without rounded borders, add the .**jumbotron-fluid** class and a .**container** or .**container-fluid** inside of it.

Look at 04-CSS-Frameworks/06-Jumbotron/index2.html

**Images [**<https://www.w3schools.com/bootstrap4/bootstrap_images.asp>**]**

The .**rounded** class adds rounded corners to an image.

The .**rounded-circle** class shapes the image to a circle.

The .**img-thumbnail** class shapes the image to a thumbnail (bordered):

Look at 04-CSS-Frameworks/07-Images/index.html

**Lists [**<https://www.w3schools.com/bootstrap4/bootstrap_list_groups.asp>**]**

To create a basic list group, use an <ul> element with class .**list-group**, and <li> elements with class .**list-group-item**. Use the .**active** class to highlight the current item. By default, the list-group occupies the full width of the container it is in. If you want the list to occupy less space, place it inside a row and specify its width using “col” classes.

Look at 04-CSS-Frameworks/08-Lists/index1.html

To create a list group with linked items, use <div> instead of <ul>, and <a> instead of <li>. Optionally, add the .**list-group-item-action** class if you want a grey background color on hover.

The .**disabled** class adds a lighter text color to the disabled item. And when used on links, it will remove the hover effect.

Look at 04-CSS-Frameworks/08-Lists/index2.html

Use the .**list-group-flush** class to remove some borders and rounded corners.

Look at 04-CSS-Frameworks/08-Lists/index3.html

To display horizontally instead of vertically (side by side instead of on top of each other), add the .**list-group-horizontal** class to .**list-group**.

Look at 04-CSS-Frameworks/08-Lists/index4.html

Contextual classes, e.g., **list-group-item-primary**, **list-group-item-success** etc. can be used to color list items.

Look at 04-CSS-Frameworks/08-Lists/index5.html

Combine .**badge** classes with utility/helper classes to add badges inside the list group.

Look at 04-CSS-Frameworks/08-Lists/index6.html

**Tables [**<https://www.w3schools.com/bootstrap4/bootstrap_tables.asp>**]**

A basic Bootstrap table has a light padding and horizontal dividers. The .**table** class adds basic styling to a table.

The .**table-hover** class adds a hover effect (grey background color) on table rows.

Look at 04-CSS-Frameworks/09-Tables/index1.html

The .**table-striped** class adds zebra-stripes to a table.

Look at 04-CSS-Frameworks/09-Tables/index2.html

The .**table-bordered** class adds borders on all sides of the table and cells.

Look at 04-CSS-Frameworks/09-Tables/index3.html

The .**table-dark** class adds a black background to the table.

Look at 04-CSS-Frameworks/09-Tables/index4.html

The .**table-borderless** class removes borders from the table.

Contextual classes, e.g., **table-primary**, **table-success** etc., can be used to color the whole table (<**table**>), the table rows (<**tr**>) or table cells (<**td**>).

Look at 04-CSS-Frameworks/09-Tables/index5.htm

The .**thead-dark** class adds a black background to table headers, and the .**thead-light** class adds a grey background to table headers.

The .**table-sm** class makes the table smaller by cutting cell padding in half.

Look at 04-CSS-Frameworks/09-Tables/index6.htm

The .**table-responsive** class creates a responsive table: an horizontal scrollbar is added to the table on screens that are less than 992px wide (if needed). When viewing on anything larger than 992px wide, there is no difference. You can also decide when the table should get a scrollbar, depending on screen width: .**table-responsive-sm**, .**table-responsive-md**, etc.

Look at 04-CSS-Frameworks/09-Tables/index7.html

**Buttons [**<https://www.w3schools.com/bootstrap4/bootstrap_buttons.asp>**]**

Bootstrap provides different styles of buttons. The button classes can be used on <a>, <button>, or <input> elements:

|  |
| --- |
| <button type="button" class="btn">Basic</button>  <button type="button" class="btn btn-primary">Primary</button>  <a href="#" class="btn btn-info" role="button">Link Button</a>0  <input type="button" class="btn btn-info" value="Input Button">  <input type="submit" class="btn btn-info" value="Submit Button"> |

You can also have outline/bordered buttons:

|  |
| --- |
| <button type="button" class="btn btn-outline-primary">Primary</button>  <button type="button" class="btn btn-outline-secondary">Secondary</button>  <button type="button" class="btn btn-outline-success">Success</button> |

Use the .**btn-lg** class for large buttons or .**btn-sm** class for small buttons:

|  |
| --- |
| <button type="button" class="btn btn-primary btn-lg">Large</button>  <button type="button" class="btn btn-primary">Default</button>  <button type="button" class="btn btn-primary btn-sm">Small</button> |

Add class .**btn-block** to create a block level button that spans the entire width of the parent element.

|  |
| --- |
| <button type="button" class="btn btn-primary btn-block">Full-Width Button</button> |

A button can be set to an active (appear pressed) or a disabled (unclickable) state. The class .active makes a button appear pressed, and the disabled attribute makes a button unclickable. Note that <a> elements do not support the disabled attribute and must therefore use the .disabled class to make it visually appear disabled.

|  |
| --- |
| <button type="button" class="btn btn-primary active">Active Primary</button>  <button type="button" class="btn btn-primary" disabled>Disabled Primary</button>  <a href="#" class="btn btn-primary disabled">Disabled Link</a> |

Look at 04-CSS-Frameworks/10-Buttons/index1.html

Bootstrap 4 allows you to group a series of buttons together (on a single line) in a button group. Use a <div> element with class .**btn-group** to create a button group:

|  |
| --- |
| <div class="btn-group">  <button type="button" class="btn btn-primary">Apple</button>  <button type="button" class="btn btn-primary">Samsung</button>  <button type="button" class="btn btn-primary">Sony</button>  </div> |

Bootstrap also supports vertical button groups. Use the class .**btn-group-vertical** to create a vertical button group.

|  |
| --- |
| <div class="btn-group-vertical">  <button type="button" class="btn btn-primary">Apple</button>  <button type="button" class="btn btn-primary">Samsung</button>  <button type="button" class="btn btn-primary">Sony</button>  </div> |

Look at 04-CSS-Frameworks/10-Buttons/index2.html

**Forms [**<https://www.w3schools.com/bootstrap4/bootstrap_forms.asp>**]**

Bootstrap provides two types of form layouts:

* Stacked (full-width) form
* Inline form

In a stacked form, all textual <input>, <textarea>, and <select> elements with class .**form-control** have a width of 100%. In other words, they occupy the full width of the container. We also have to add a wrapper element with .**form-group**, around each form control, to ensure proper margins.

In an inline form however, all form elements are laid out side-by-side with their default widths and all fit into a single row of the container. If the elements do not fit on a single line, then they wrap to the next line. Add class .**form-inline** to the <form> element.

Look at 04-CSS-Frameworks/11-Forms/index1.html

In a stacked form, all form-elements occupy the full width of the container. But it is possible to put two form element side-by-side on the same row and make them responsive using .col container. Also remember to put all elements on the same row inside a .row container. If you want less grid margins (override default column gutters), use .**form-row** instead of .**row**

Look at 04-CSS-Frameworks/11-Forms/index2.html

**Supported Form Controls**

Bootstrap supports the following form controls:

* Input (text, password, datetime, datetime-local, date, month, time, week, number, email, url, search, tel, color)
* textarea
* checkbox
* radio
* select

Bootstrap supports all HTML5 **input** types: text, password, datetime, datetime-local, date, month, time, week, number, email, url, search, tel, and color. We have already seen email and password input fields in the previous two examples. Inputs will NOT be fully styled if their type is not properly declared.

The following example contains a **textarea**.

Look at 04-CSS-Frameworks/11-Forms/index3.html

**Checkboxes** are used if you want the user to select any number of options from a list of preset options. The following example contains three checkboxes. The last option is disabled. Use a wrapper element with class="**form-check**" to ensure proper margins for labels and checkboxes. Add the .**form-check-label** class to label elements, and .**form-check-input** to style checkboxes properly inside the .**form-check** container. For inline checkboxes, i.e., checkboxes that get placed side-by-side on the same line, use the .**form-check-inline** class.

Look at 04-CSS-Frameworks/11-Forms/index4.html

**Radio buttons** are used if you want to limit the user to just one selection from a list of preset options. The following example contains three radio buttons. The last option is disabled. As with checkboxes, use the .**form-check-inline** class if you want the radio buttons to appear on the same line.

Look at 04-CSS-Frameworks/11-Forms/index5.html

**Select lists** are used if you want to allow the user to pick from multiple options. The following example contains a dropdown list (select list).

Look at 04-CSS-Frameworks/11-Forms/index6.html

You can change the size of the form control with .**form-control-sm** or .**form-control-lg**.

Use the .**form-control-plaintext** if you want to style the input field as plain text.

Add the .**form-control-range** class to input type"range" or .form-control-file to input type"file" to style a range control or a file field with full-width.

Look at 04-CSS-Frameworks/11-Forms/index7.html

**Form Validation**

You can use different validation classes to provide valuable feedback to users. Add either .**was-validated** or .**needs-validation** to the <form> element, depending on whether you want to provide validation feedback before or after submitting the form. The input fields will have a green (valid) or red (invalid) border to indicate what's missing in the form. You can also add a .**valid-feedback** or .**invalid-feedback** message to tell the user explicitly what's missing, or needs to be done before submitting the form.

Look at 04-CSS-Frameworks/11-Forms/index8.html

In the following example, we use .**needs-validation**, which will add the validation effect AFTER the form has been submitted. That is, when the user presses the Submit button, the validation code written in JQuery is called, which determines whether what is entered by the user for the input fields of the form are valid or not. If any one of the input field is invalid, then the warning message is displayed. When all fields are correctly validated, then the form us submitted to the link specified in the form’s action property, which is this page in this example. *Note that you will also have to add some jQuery code for this example to work properly*.

Look at 04-CSS-Frameworks/11-Forms/index9.html

**Dropdown Menus [**<https://www.w3schools.com/bootstrap4/bootstrap_dropdowns.asp>**]**

A dropdown menu is a toggleable menu that allows the user to choose one value from a predefined list. The .**dropdown** class indicates a dropdown menu. To open the dropdown menu, use a button or a link with a class of .**dropdown-toggle** and the **data-toggle="dropdown"** attribute. Add the .**dropdown-menu** class to a <div> element to actually build the dropdown menu. Then add the .**dropdown-item** class to each element (links or buttons) inside the dropdown menu. If you want the dropdown menu to expand upwards instead of downwards, change the <div> element with class="dropdown" to "dropup".

Look at 04-CSS-Frameworks/12-Dropdowns/index1.html

It is possible to have a group of buttons where each button is associated with a dropdown menu.

Look at 04-CSS-Frameworks/12-Dropdowns/index2.html

**Navigation [**<https://www.w3schools.com/bootstrap4/bootstrap_navs.asp>**]**

If you want to create a simple horizontal menu, add the .**nav** class to a <ul> element, followed by .**nav-item** for each <li> and add the .**nav-link** class to their links.

Look at 04-CSS-Frameworks/13-Navigation/index1.html

Add the .**flex-column** class to create a vertical nav.

Look at 04-CSS-Frameworks/13-Navigation/index2.html

You can turn the nav menu into navigation tabs with the .**nav-tabs** class. Add the .**active** class to the active/current link. If you want the tabs to be togglable, see the last example on this page. To make the tabs toggleable, add the **data-toggle="tab"** attribute to each link. Then add a .**tab-pane** class with a unique ID for every tab and wrap them inside a <div> element with class .**tab-content**. If you want the tabs to fade in and out when clicking on them, add the .**fade** class to .**tab-pane.**

Look at 04-CSS-Frameworks/13-Navigation/index3.html

**Navigation Bars [**<https://www.w3schools.com/bootstrap4/bootstrap_navbar.asp>**]**

A navigation bar is a navigation header that is placed at the top of the page. A standard navigation bar is created with the .**navbar** class, followed by a responsive collapsing class: .**navbar-expand-xl**|lg|md|sm (stacks the navbar vertically on extra large, large, medium or small screens). To add links inside the navbar, use a <ul> element with class="navbar-nav". Then add <li> elements with a .nav-item class followed by an <a> element with a .nav-link class.

Look at 04-CSS-Frameworks/14-Navbars/index1.html

Very often, especially on small screens, you want to hide the navigation links and replace them with a button that should reveal them when clicked on. To create a collapsible navigation bar, use a button with **class="navbar-toggler**", **data-toggle="collapse"** and **data-target="#thetarget"**. Then wrap the navbar content (links, etc) inside a div element with **class="collapse navbar-collapse"**, followed by an id that matches the data-target of the button: **"thetarget"**.

Look at 04-CSS-Frameworks/14-Navbars/index2.html

Navbars can also hold dropdown menus and forms.

Look at 04-CSS-Frameworks/14-Navbars/index3.html

**Alerts [**<https://www.w3schools.com/bootstrap4/bootstrap_alerts.asp>**]**

Bootstrap provides an easy way to create predefined alert messages. Alerts are created with the .**alert** class, followed by one of the contextual classes .**alert-success**, .**alert-info**, .**alert-warning**, .**alert-danger**, .**alert-primary**, .**alert-secondary**, .**alert-light** or .**alert-dark**. It is possible to close the alert message by adding a .**alert-dismissible** class to the alert container. Then add **class="close"** and **data-dismiss="alert"** to a link or a button element (when you click on this the alert box will disappear). The .**fade** and .**show** classes adds a fading effect when closing the alert message.

Look at 04-CSS-Frameworks/15-Alerts/index.html

**Carousel [**<https://www.w3schools.com/bootstrap4/bootstrap_carousel.asp>**]**

The Carousel is a slideshow for cycling through elements. Many Web pages use a carousel to cycle through a set of images.

Look at 04-CSS-Frameworks/16-Carousel/index.html

**Modals [**<https://www.w3schools.com/bootstrap4/bootstrap_modal.asp>**]**

The Modal component is a dialog box/popup window that is displayed on top of the current page. Use the .**fade** class to add a fading effect when opening and closing the modal. Center the modal vertically and horizontally within the page, with the .**modal-dialog-centered** class. Also note that modals require JS libraries to be included.

Look at 04-CSS-Frameworks/17-Modals/index.html

**Badges [**<https://www.w3schools.com/bootstrap4/bootstrap_badges.asp>**]**

Badges are used to add additional information to any content. Use the .**badge** class together with a contextual class (like .**badge-secondary**) *within <span> elements* to create rectangular badges. Note that badges scale to match the size of the parent element (if any).

Use any of the contextual classes (.badge-\*) to change the color of a badge.

Use the .**badge-pill** class to make the badges more round.

It is possible to use a badge inside a button if you enclose it inside a span element.

It is possible to combine .**badge** classes with utility/helper classes to add badges inside a list group.

Look at 04-CSS-Frameworks/18-Badges/index.html

**Progress Bars [**<https://www.w3schools.com/bootstrap4/bootstrap_progressbars.asp>**]**

A progress bar can be used to show a user how far along he/she is in a process. To create a default progress bar, add a .**progress** class *to a container element* and add the .**progress-bar** class *to its child element*. Use the CSS width property to set the width of the progress bar.

The height of the progress bar is 16px by default. Use the CSS height property to change it. Note that you must set the same height for the progress container and the progress bar.

It is possible to add text inside the progress bar to show the visible percentage.

By default, the progress bar is blue (primary). Use any of the Bootstrap contextual background classes to its color.

Use the .**progress-bar-striped** class to add stripes to the progress bars.

Add the .**progress-bar-animated** class to animate the progress bar.

Progress bars can also be stacked.

Look at 04-CSS-Frameworks/19-ProgressBars/index.html

**Spinners [**<https://www.w3schools.com/bootstrap4/bootstrap_spinners.asp>**]**

To create a spinner/loader, use the .spinner-border class:

|  |
| --- |
| <div class="spinner-border"></div> |

Use any text color utilities to add a color to the spinner:

|  |
| --- |
| <div class="spinner-border text-muted"></div>  <div class="spinner-border text-primary"></div> |

Use the .spinner-grow class if you want the spinner/loader to grow instead of "spin":

|  |
| --- |
| <div class="spinner-grow text-muted"></div>  <div class="spinner-grow text-primary"></div> |

Use .spinner-border-sm or .spinner-grow-sm to create a smaller spinner:

|  |
| --- |
| <div class="spinner-border spinner-border-sm"></div>  <div class="spinner-grow spinner-grow-sm"></div> |

You can also add "spinners" to a button:

|  |
| --- |
| <button class="btn btn-primary">  <span class="spinner-border spinner-border-sm"></span>  </button>  <button class="btn btn-primary">  <span class="spinner-border spinner-border-sm"></span>  Loading..  </button> |

Look at 04-CSS-Frameworks/20-Spinners/index.html

**Floats & Flexbox**

In some cases it might be necessary to float an element to the left or right inside a container. Similarly, it may be necessary to use flexbox to order and layout HTML elements inside a container.

You can use “**float-right**” class to float the element to the right. “**float-left**” class floats it to the left. Recall though that you must clear the floats at the end as you remember. To do this, put the elements being floated inside a container that also belongs to the “**clearfix**” class.

With flexbox, you can use all of flexbox’s properties such as fill, shrink, order, justify etc. For more details look at the following page: <https://www.w3schools.com/bootstrap4/bootstrap_flex.asp>

Look at 04-CSS-Frameworks/21-Float/index.html

**Other Bootstrap Utilities [**<https://www.w3schools.com/bootstrap4/bootstrap_utilities.asp>**]**

Bootstrap has a lot of utility/helper classes to quickly style elements without using any CSS code. These include classes for borders, border colors, border shapes, padding, margins, alignment, element width, height, colors, icons, shadows, etc. For a complete list of Bootstrap utilities and how you can use them, look at the link given in the section title.

**Other Bootstrap Components [**<https://getbootstrap.com/docs/4.4/components/>**]**

In addition to the w3schools web page, you can go to Bootstrap’s official web page (link given at the section title) and look at how each component is used.

**Bootstrap 5 Tutorial by Net Ninja**

The current version of bootstrap is 5. A tutorial on bootstrap 5 by Net Ninja can be found here:

<https://www.youtube.com/playlist?list=PL4cUxeGkcC9joIM91nLzd_qaH_AimmdAR>

**Materialize CSS [**https://materializecss.com/**]**

Created and designed by Google, Material Design is a design language that combines the classic principles of successful design along with innovation and technology. Google's goal is to develop a system of design that allows for a unified user experience across all their products on any platform. Materialize is very similar to Bootstrap in the sense that it is a large library that includes many ready higher level components that you can directly use to develop your page.

Here is a tutorial on Materialize CSS by Net Ninja.

<https://www.youtube.com/watch?v=gCZ3y6mQpW0&list=PL4cUxeGkcC9gGrbtvASEZSlFEYBnPkmff>

The following examples are taken from this tutorial:

1. Container & Headings: 04-CSS-Frameworks/22-Materialize/index1.html
2. Colors: 04-CSS-Frameworks/22-Materialize/index2.html
3. Buttons: 04-CSS-Frameworks/22-Materialize/index3.html
4. Icons: 04-CSS-Frameworks/22-Materialize/index4.html
5. Grid System: 04-CSS-Frameworks/22-Materialize/index5.html
6. Shadows & Depth: 04-CSS-Frameworks/22-Materialize/index6.html
7. Navbar: 04-CSS-Frameworks/22-Materialize/index7.html
8. Badges: 04-CSS-Frameworks/22-Materialize/index8.html
9. Cards: 04-CSS-Frameworks/22-Materialize/index9.html
10. Collections: 04-CSS-Frameworks/22-Materialize/index10.html
11. Modals: 04-CSS-Frameworks/22-Materialize/index11.html
12. Forms: 04-CSS-Frameworks/22-Materialize/index12.html
13. A complete responsive Web page with Material components by Net Ninja:

04-CSS-Frameworks/22-Materialize/index13.html

**Bulma CSS [**https://bulma.io/**]**

Another big CSS framework similar to Bootstrap and Material is Bulma CSS. It comes with a lot of ready-to-use classes and components. Here is a tutorial on Bulma CSS by Net Ninja.

<https://www.youtube.com/watch?v=SCSAExGFK1E&list=PL4cUxeGkcC9iXItWKbaQxcyDT1u6E7a8a>

**Tailwind CSS [**https://tailwindcss.com/**]**

Tailwind is a CSS framework that is made up of many utility classes that you can combine to create your own custom components. In this sense, Tailwind CSS is much lower-level than Bootstap, Materialize, Bulma, etc. in that rather than giving you higher level components such as cards, navbars, collections, etc., tailwind gives you lower-level utility classes that you can use to create your own custom components easily. It is a tradeoff in the sense that you need to combine many utility classes to create your own components, but it is much more flexible than the other larger CSS frameworks.

Here is a tutorial on Tailwind CSS by Net Ninja.

<https://www.youtube.com/watch?v=bxmDnn7lrnk&list=PL4cUxeGkcC9gpXORlEHjc5bgnIi5HEGhw>

Tailwind also has a Just-in-Time compiler option that you can use. Here is a tutorial on Tailwind Just in Time:

<https://www.youtube.com/watch?v=aQS7kaje-24&list=PL4cUxeGkcC9ht1OMQPhBVKAb2dVLhg-MJ>

If you want to learn Tailwind CSS directly from the developers, here is their youtube channel:

<https://www.youtube.com/c/TailwindLabs/videos>

**Using Tailwind CSS**

It is possible to make use of Tailwind CSS using a complete .css file from a CDN as we do with bootstap and materialize:

04-CSS-Frameworks/23-Tailwind/public/index1.html

The problem with this method is that even though you may only use a very small subset of features from tailwind, we make the browser download the entire Tailwind css style file, which is usually big.

An alternative and recommended way to use Tailwind is to use the CLI (<https://tailwindcss.com/docs/installation>).

1. First, install tailwindcss using npm and then create a “tailwind.config.js” file as follows:

% npm install -D tailwindcss

% npx tailwindcss init

1. Add the paths to all of your template files in your tailwind.config.js file. Notice that we want to feed in all html and javascript files to tailwindcss.

/\*\* @type {import('tailwindcss').Config} \*/

module.exports = {

    content: ["\*.{html,js}"],

    theme: {

        extend: {},

    },

    plugins: [],

}

1. Add the Tailwind directives to your input.css file:

@tailwind base;

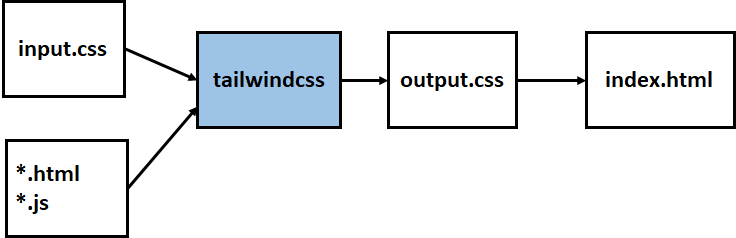
@tailwind components;

@tailwind utilities;

1. Start the Tailwind CLI build process:

% npx tailwindcss -i ./input.css -o ./output.css –watch

Here we run tailwindcss module feeding input.css as the input. Tailwindcss uses the “content: ["\*.{html,js}"]” line from tailwind.config.js, which specifies that tailwindcss needs to look into all html and javascript files to create output.css that contains ONLY the necessary css rules. You then include output.css in your index.html file. This is illustrated below:



For the sake of simplicity, I will be using the CDN version of the tailwind in the following examples. The entire documentation for Tailwind can be found here: <https://tailwindcss.com/docs/>

Go here, and look at all classes you can use in your html file. Here are some examples:

1. Fonts and colors: 04-CSS-Frameworks/23-Tailwind/index1.html
2. Padding, margins, width: 04-CSS-Frameworks/23-Tailwind/index2.html
3. Grid: 04-CSS-Frameworks/23-Tailwind/index3.html

**UI/UX Design**

Laying items on an HTML page requires artistry and is usually done by graphic designers. A great Web site on good layout practices is the one by Jen Simmons, who is a Mozilla developer: <https://labs.jensimmons.com/>. She also has a YouTube channel where she has many videos on good layout choices and how to implement them using CSS grid & flex box: <https://www.youtube.com/c/LayoutLand/videos>.

Because she is on the CSS working group that designed the CSS grid specs, she advocates using CSS grid for responsive Web pages and is kind of against the common practice of specifying the exact screen layout at different screen sizes (break-points) using media queries. Instead, she advocates using the following CSS grid declaration to fit as many items onto a row to make the design responsive:

SOME CONTAINER OBJECT {

display: grid;

grid-template-columns: repeat(auto-fit, minmax(300px, 1fr));

grid-gap: 1rem;

}

Here is a page where she uses this idea to fit as many card objects onto a responsive grid: <https://labs.jensimmons.com/2017/03-009.html>. To design a single card, she uses the “flex” layout manager to vertically place the items. Here is another Web page where she uses the same idea: <https://labs.jensimmons.com/2016/examples/spices-1.html>.

Here is another example where she uses the same idea but with the “grid-auto-flow: dense;” option enabled, which makes items to move to previous rows to fill gaps. <https://labs.jensimmons.com/2016/examples/image-gallery-grid-1.html>. Here is the link to a video where she explains the design details: <https://www.youtube.com/watch?v=qNtJ5p3h2A4>.

I strongly recommend that you go over all videos on Jen Simmons’s channel to get a better idea on responsive Web page design using CSS grid and different layout techniques.