MODULE 4

WEB DESIGN with CLIENT SIDE SCRIPTING

FEU ALABANG FEU DILIMAN FEU TECH

Technology Driven by Innovation

SUB TOPIC 2

Bootstrap Framework Overview

Bootstrap - Overview

Components and options for laying out your Bootstrap project, including wrapping containers, a powerful grid system, a flexible media object, and responsive utility classes.

Bootstrap - Overview

Containers

Containers are the most basic layout element in Bootstrap and are required when using our default grid system. Containers are used to contain, pad, and (sometimes) center the content within them. While containers can be nested, most layouts do not require a nested container.

Bootstrap comes with three different containers:

.container, which sets a max-width at each responsive breakpoint
.container-fluid, which is width: 100% at all breakpoints
.container-{breakpoint}, which is width: 100% until the specified breakpoint

Bootstrap - Overview

All-in-one
Our default .container class is a responsive, fixed-width container, meaning its max-width changes at each breakpoint.

Fluid
Use .container-fluid for a full width container, spanning the entire width of the viewport.

Responsive
Responsive containers are new in Bootstrap v4.4. They allow you to specify a class that is 100% wide until the specified breakpoint is reached, after which we apply max-widths for each of the higher breakpoints. For example, .container-sm is 100% wide to start until the sm breakpoint is reached, where it will scale up with md, lg, and xl.

Bootstrap - Overview

Responsive breakpoints
Since Bootstrap is developed to be mobile first, we use a handful of media queries to create sensible breakpoints for our layouts and interfaces. These breakpoints are mostly based on minimum viewport widths and allow us to scale up elements as the viewport changes.

Bootstrap primarily uses the following media query ranges—or breakpoints—in our source Sass files for our layout, grid system, and components.

Z-index
Several Bootstrap components utilize z-index, the CSS property that helps control layout by providing a third axis to arrange content. We utilize a default z-index scale in Bootstrap that's been designed to properly layer navigation, tooltips and popovers, modals, and more.

These higher values start at an arbitrary number, high and specific enough to ideally avoid conflicts. We need a standard set of these across our layered components—tooltips, popovers, navbars, dropdowns, modals—so we can be reasonably consistent in the behaviors. There's no reason we couldn't have used 100+ or 500+.

We don't encourage customization of these individual values; should you change one, you likely need to change them all.

Bootstrap - Overview

Reboot
Reboot, a collection of element-specific CSS changes in a single file, kickstart Bootstrap to provide an elegant, consistent, and simple baseline to build upon.

Approach
Reboot builds upon Normalize, providing many HTML elements with somewhat opinionated styles using only element selectors. Additional styling is done only with classes. For example, we reboot some <table> styles for a simpler baseline and later provide .table, .table-bordered, and more.

Here are our guidelines and reasons for choosing what to override in Reboot:

Update some browser default values to use rems instead of ems for scalable component spacing.
Avoid margin-top. Vertical margins can collapse, yielding unexpected results. More importantly though, a single direction of margin is a simpler mental model.
For easier scaling across device sizes, block elements should use rems for margins.
Keep declarations of font-related properties to a minimum, using inherit whenever possible.

Bootstrap - Overview

Page defaults
The <html> and <body> elements are updated to provide better page-wide defaults. More specifically:

The box-sizing is globally set on every element—including *::before and *::after, to border-box. This ensures that the declared width of element is never exceeded due to padding or border.
No base font-size is declared on the <html>, but 16px is assumed (the browser default).
font-size: 1rem is applied on the <body> for easy responsive type-scaling via media queries while respecting user preferences and ensuring a more accessible approach.
The <body> also sets a global font-family, line-height, and text-align. This is inherited later by some form elements to prevent font inconsistencies.
For safety, the <body> has a declared background-color, defaulting to #fff.

Bootstrap - Overview

Native font stack

The default web fonts (Helvetica Neue, Helvetica, and Arial) have been dropped in Bootstrap 4 and replaced with a “native font stack” for optimum text rendering on every device and OS. Read more about native font stacks in this Smashing Magazine article.

Headings and paragraphs

All heading elements—e.g., <h1>—and <p> are reset to have their margin-top removed. Headings have margin-bottom: .5rem added and paragraphs margin-bottom: 1rem for easy spacing.

Bootstrap - Overview

Forms

Various form elements have been rebooted for simpler base styles. Here are some of the most notable changes:

- <fieldset>s have no borders, padding, or margin so they can be easily used as wrappers for individual inputs or groups of inputs.
- <legend>s, like fieldsets, have also been restyled to be displayed as a heading of sorts.
- <label>s are set to display: inline-block to allow margin to be applied.
- <input>s, <select>s, <textarea>s, and <button>s are mostly addressed by Normalize, but Reboot removes their margin and sets line-height: inherit, too.
- <textarea>s are modified to only be resizable vertically as horizontal resizing often “breaks” page layout.
- <button>s and <input> button elements have cursor: pointer when :not(:disabled).

These changes, and more, are demonstrated below.

Bootstrap - Overview

Alerts
Provide contextual feedback messages for typical user actions with the handful of available and flexible alert messages.

Badges
Documentation and examples for badges, our small count and labeling component.

Breadcrumb
Indicate the current page's location within a navigational hierarchy that automatically adds separators via CSS.

Bootstrap - Overview

Buttons
Use Bootstrap’s custom button styles for actions in forms, dialogs, and more with support for multiple sizes, states, and more.

Button group
Group a series of buttons together on a single line with the button group, and super-power them with JavaScript.

Cards
Bootstrap’s cards provide a flexible and extensible content container with multiple variants and options.

Bootstrap - Overview

Carousel
A slideshow component for cycling through elements—images or slides of text—like a carousel.

Collapse
Toggle the visibility of content across your project with a few classes and our JavaScript plugins.

Borders
Use border utilities to quickly style the border and border-radius of an element. Great for images, buttons, or any other element.

Clearfix
Quickly and easily clear floated content within a container by adding a clearfix utility.

Bootstrap - Overview

Approach
Learn about the guiding principles, strategies, and techniques used to build and maintain Bootstrap so you can more easily customize and extend it yourself.

While the getting started pages provide an introductory tour of the project and what it offers, this document focuses on why we do the things we do in Bootstrap. It explains our philosophy to building on the web so that others can learn from us, contribute with us, and help us improve.

See something that doesn’t sound right, or perhaps could be done better? Open an issue—we’d love to discuss it with you.

Bootstrap - Overview

Responsive
Bootstrap's responsive styles are built to be responsive, an approach that's often referred to as mobile-first. We use this term in our docs and largely agree with it, but at times it can be too broad. While not every component must be entirely responsive in Bootstrap, this responsive approach is about reducing CSS overrides by pushing you to add styles as the viewport becomes larger.

Across Bootstrap, you'll see this most clearly in our media queries. In most cases, we use min-width queries that begin to apply at a specific breakpoint and carry up through the higher breakpoints. For example, a .d-none applies from min-width: 0 to infinity. On the other hand, a .d-md-none applies from the medium breakpoint and up.

At times we'll use max-width when a component's inherent complexity requires it. At times, these overrides are functionally and mentally clearer to implement and support than rewriting core functionality from our components. We strive to limit this approach, but will use it from time to time.

Bootstrap - Overview

Classes
Aside from our Reboot, a cross-browser normalization stylesheet, all our styles aim to use classes as selectors. This means steering clear of type selectors (e.g., input[type="text"]) and extraneous parent classes (e.g., .parent .child) that make styles too specific to easily override.

As such, components should be built with a base class that houses common, not-to-be overridden property-value pairs. For example, .btn and .btn-primary. We use .btn for all the common styles like display, padding, and border-width. We then use modifiers like .btn-primary to add the color, background-color, border-color, etc.

Modifier classes should only be used when there are multiple properties or values to be changed across multiple variants. Modifiers are not always necessary, so be sure you're actually saving lines of code and preventing unnecessary overrides when creating them. Good examples of modifiers are our theme color classes and size variants.

Bootstrap - Overview

z-index scales
There are two z-index scales in Bootstrap—elements within a component and overlay components.

Component elements
- Some components in Bootstrap are built with overlapping elements to prevent double borders without modifying the border property. For example, button groups, input groups, and pagination.
- These components share a standard z-index scale of 0 through 3.
- 0 is default (initial), 1 is :hover, 2 is :active/.active, and 3 is :focus.
- This approach matches our expectations of highest user priority. If an element is focused, it’s in view and at the user’s attention. Active elements are second highest because they indicate state. Hover is third highest because it indicates user intent, but nearly anything can be hovered.

Bootstrap - Overview

Overlay components
Bootstrap includes several components that function as an overlay of some kind. This includes, in order of highest z-index, dropdowns, fixed and sticky navbars, modals, tooltips, and popovers. These components have their own z-index scale that begins at 1000. This starting number was chosen arbitrarily and serves as a small buffer between our styles and your project’s custom styles.

Each overlay component increases its z-index value slightly in such a way that common UI principles allow user focused or hovered elements to remain in view at all times. For example, a modal is document blocking (e.g., you cannot take any other action save for the modal’s action), so we put that above our navbars.

Learn more about this in our z-index layout page.

Bootstrap - Overview

HTML and CSS over JS
Whenever possible, we prefer to write HTML and CSS over JavaScript. In general, HTML and CSS are more prolific and accessible to more people of all different experience levels. HTML and CSS are also faster in your browser than JavaScript, and your browser generally provides a great deal of functionality for you.

This principle is our first-class JavaScript API using data attributes. You don't need to write nearly any JavaScript to use our JavaScript plugins; instead, write HTML. Read more about this in our JavaScript overview page.

Lastly, our styles build on the fundamental behaviors of common web elements. Whenever possible, we prefer to use what the browser provides. For example, you can put a .btn class on nearly any element, but most elements don't provide any semantic value or browser functionality. So instead, we use <button>s and <a>s.

The same goes for more complex components. While we could write our own form validation plugin to add classes to a parent element based on an input's state, thereby allowing us to style the text say red, we prefer using the :valid/:invalid pseudo-elements every browser provides us.

Bootstrap - Overview

Utilities
Utility classes—formerly helpers in Bootstrap 3—are a powerful ally in combatting CSS bloat and poor page performance. A utility class is typically a single, immutable property-value pairing expressed as a class (e.g., .d-block represents display: block;). Their primary appeal is speed of use while writing HTML and limiting the amount of custom CSS you have to write.

Specifically regarding custom CSS, utilities can help combat increasing file size by reducing your most commonly repeated property-value pairs into single classes. This can have a dramatic effect at scale in your projects.

Flexible HTML
While not always possible, we strive to avoid being overly dogmatic in our HTML requirements for components. Thus, we focus on single classes in our CSS selectors and try to avoid immediate children selectors (>). This gives you more flexibility in your implementation and helps keep our CSS simpler and less specific.

Bootstrap - Overview

Migrating to v4
Bootstrap 4 is a major rewrite of the entire project. The most notable changes are summarized below, followed by more specific changes to relevant components.

Stable changes
Moving from Beta 3 to our stable v4.x release, there are no breaking changes, but there are some notable changes.

Printing
Fixed broken print utilities. Previously, using a .d-print-* class would unexpectedly overrule any other .d-* class. Now, they match our other display utilities and only apply to that media (@media print).

Expanded available print display utilities to match other utilities. Beta 3 and older only had block, inline-block, inline, and none. Stable v4 added flex, inline-flex, table, table-row, and table-cell.

Fixed print preview rendering across browsers with new print styles that specify @page size.

Bootstrap - Overview

Beta 3 changes

While Beta 2 saw the bulk of our breaking changes during the beta phase, but we still have a few that needed to be addressed in the Beta 3 release. These changes apply if you're updating to Beta 3 from Beta 2 or any older version of Bootstrap.

Miscellaneous

- Removed the unused $thumbnail-transition variable. We weren't transitioning anything, so it was just extra code.
- The npm package no longer includes any files other than our source and dist files; if you relied on them and were running our scripts via the node_modules folder, you should adapt your workflow.

Bootstrap - Overview

Forms

- Rewrote both custom and default checkboxes and radios. Now, both have matching HTML structure (outer <div> with sibling <input> and <label>) and the same layout styles (stacked default, inline with modifier class). This allows us to style the label based on the input's state, simplifying support for the disabled attribute (previously requiring a parent class) and better supporting our form validation.

As part of this, we've changed the CSS for managing multiple background-images on custom form checkboxes and radios. Previously, the now removed .custom-control-indicator element had the background color, gradient, and SVG icon. Customizing the background gradient meant replacing all of those every time you needed to change just one. Now, we have .custom-control-label::before for the fill and gradient and .custom-control-label::after handles the icon.

Bootstrap - Overview

To make a custom check inline, add .custom-control-inline.

- Updated selector for input-based button groups. Instead of [data-toggle="buttons"] { } for style and behavior, we use the data attribute just for JS behaviors and rely on a new .btn-group-toggle class for styling.

- Removed .col-form-legend in favor of a slightly improved .col-form-label. This way .col-form-label-sm and .col-form-label-lg can be used on <legend> elements with ease.

- Custom file inputs received a change to their $custom-file-text Sass variable. It’s no longer a nested Sass map and now only powers one string—the Browse button as that is now the only pseudo-element generated from our Sass. The Choose file text now comes from the .custom-file-label.

Bootstrap - Overview

Input groups

- Input group addons are now specific to their placement relative to an input. We've dropped .input-group-addon and .input-group-btn for two new classes, .input-group-prepend and .input-group-append. You must explicitly use an append or a prepend now, simplifying much of our CSS. Within an append or prepend, place your buttons as they would exist anywhere else, but wrap text in .input-group-text.

- Validation styles are now supported, as are multiple inputs (though you can only validate one input per group).

- Sizing classes must be on the parent .input-group and not the individual form elements.