BOOTSTRAP

BUILD FAST & RESPONSIVE SITES

WHAT IS BOOTSTRAP?

Bootstrap is an open source product from Mark Otto and Jacob Thornton who were both employees at Twitter.

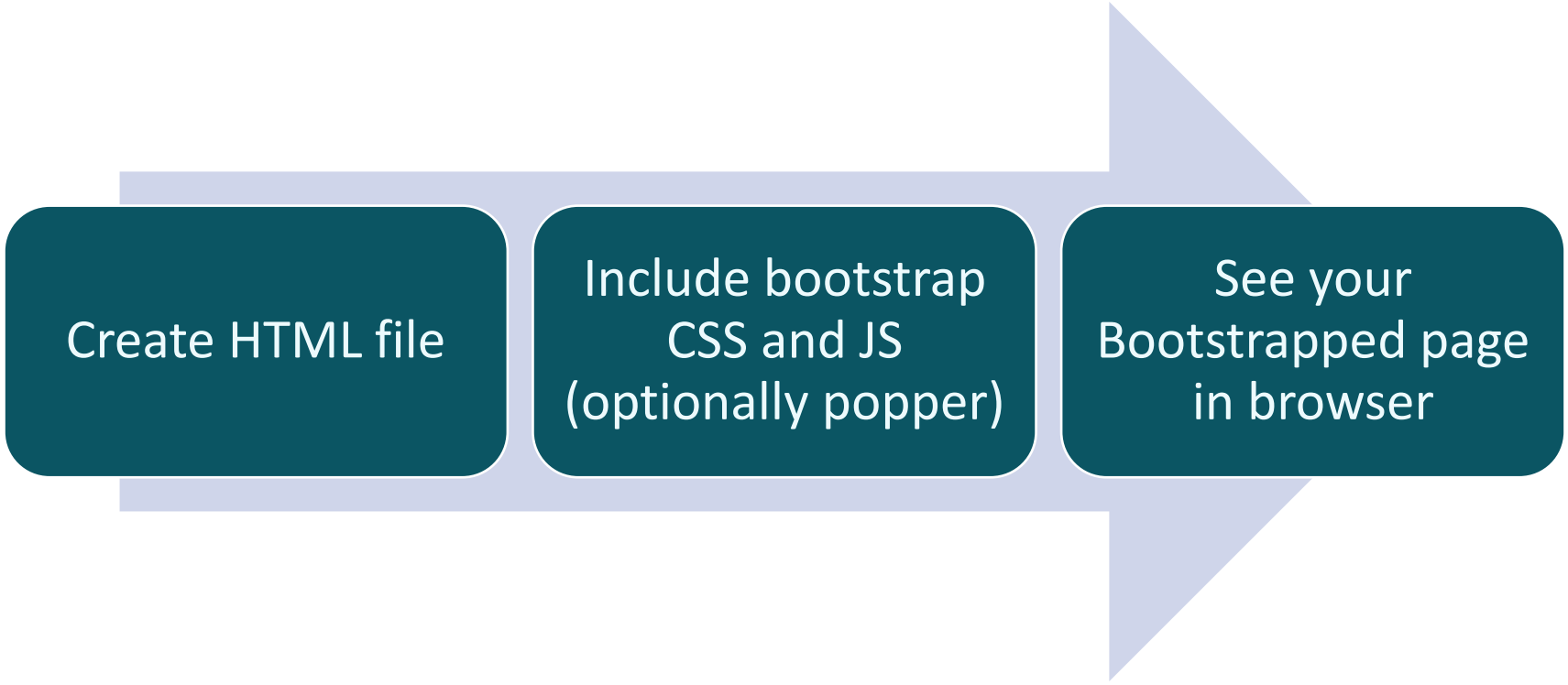
It is sleek, intuitive, and powerful mobile first front-end framework for faster and easier web development.

WHY BOOTSTRAP ?

Mobile first approach	Browser Support	Easy to get started
Responsive design	Uniform solution for building UI	Beautiful and functional built-in components

LET'S GET OUR HANDS DIRTY

You can include Bootstrap's production-ready CSS and JavaScript via CDN without the need for any build steps.



BOOTSTRAP FUNCTIONALITIES

Layout

- Breakpoints
- Container
- Grids
- Columns / Gutter

Contents

- Colors
- Typography
- Images
- Tables

Forms

- Form Controls
- Checks & Radios
- Input Groups & Floating Labels
- Form Validations

Components

- Buttons
- Alerts
- Progress Bar
- Pagination
- Navbar
- Modal

Utilities

- Vector Icons

LAYOUT : CONTAINERS

Containers are a fundamental building block of Bootstrap that contain, pad, and align your content within a given device or viewport.

Bootstrap comes with three different containers	*.container*, which sets a max-width at each responsive breakpoint
	.container-{breakpoint}, which is width: 100% until the specified breakpoint
	.container-fluid, which is width: 100% at all breakpoints

LAYOUT : GRID SYSTEM

Powerful mobile-first flexbox grid to build layouts of all shapes and sizes with twelve column system

Bootstrap’s grid system can adapt across all six default breakpoints, and any breakpoints you customize.

How the grid system comes together -	
	Grid supports six responsive breakpoints
	Containers center and horizontally pad your content
	Rows are wrappers for columns
	Columns are incredibly flexible
	Gutters are also responsive and customizable
	Sass variables, maps, and mixins power the grid

LAYOUT : BREAKPOINTS

Breakpoints are customizable widths that determine how your responsive layout behaves across device or viewport sizes in Bootstrap

Breakpoint	Class infix	Dimensions
Extra Small	None	<576px
Small	sm	≥576px
Medium	md	≥768px
Large	lg	≥992px
Extra Large	xl	≥1200px
Extra Extra Large	xxl	≥1400px

LAYOUT : COLUMNS

You can modify columns with a handful of options for alignment, ordering, and offsetting thanks to flexbox grid system.

You can also use column classes to manage widths of non-grid elements

How the Columns works -	Columns build on the grid's flexbox architecture
	When building grid layouts, all content goes in columns
	Bootstrap includes predefined classes for creating fast, responsive layouts

TYPOGRAPHY

Typography	Description
h1 to h6	Heading Classes. H1 being the highest font size.
.display	In addition to traditional headings, headings to stand out.
.lead	Make a paragraph stand out by adding .lead
blockquotes	For quoting blocks of content from another source within your document
lists	Remove the default list-style and left margin on list items (immediate children only)
.mark	will apply the same styles as <mark>
.small	will apply the same styles as <small>
.text-decoration-underline	will apply the same styles as <u>
.text-decoration-line-through	will apply the same styles as <s>

WORKING WITH IMAGES

You can use CSS Classes for opting images into responsive behavior (so they never become wider than their parent) and add lightweight styles to them

Images in Bootstrap are made responsive with .img-fluid. This applies max-width: 100%; and height: auto; to the image so that it scales with the parent width

You can use .rounded-circle to give an image a rounded border appearance

Align images with the helper float classes or text alignment classes.

WORKING WITH TABLES

Add the base class .table to any <table>, then extend with optional modifier classes or custom styles

All table styles are not inherited in Bootstrap, meaning any nested tables can be styled independent from the parent

Table Classes	Description
.table-striped	to add zebra-striping to any table row within the <tbody>
.table-striped-columns	to add zebra-striping to any table column
.table-hover	to enable a hover state on table rows within a <tbody>
.table-active	to highlight a table row or cell
.table-bordered	for borders on all sides of the table and cells
.table-responsive	to make any table responsive across all viewports

FORMS

Buttons	Use Bootstrap’s custom button styles for actions in forms, dialogs, and more with support for multiple sizes, states, and more
Alerts	Provide contextual feedback messages for typical user actions with the handful of available and flexible alert messages
Progress Bar	Custom progress bars featuring support for stacked bars, animated backgrounds, and text labels
Pagination	Pagination to indicate a series of related content exists across multiple pages
Navbar	Bootstrap’s powerful, responsive navigation header, the navbar. Includes support for branding, navigation, and more
Modal	Bootstrap’s JavaScript modal plugin to add dialogs to your site for lightboxes, user notifications, or completely custom content

BOOTSTRAP ICONS

Free, high quality, open source icon library with over 1,600 icons

Use them with or without Bootstrap in any project

```
npm i bootstrap-icons
```

SASS

MAINTAINABLE CSS

WHAT IS SASS ?

Nowadays, stylesheets are getting larger, more complex, and harder to maintain. SASS is the rescue.

Sass has features that don't exist in CSS yet like nesting, mixins, inheritance, and other goodies that help you write robust, maintainable CSS

SASS (Syntactically Awesome Stylesheet) is a CSS pre-processor, which helps to reduce repetition with CSS and saves time.

You can install Sass by downloading the package for your operating system by running the command –

```
npm install sass -g
```

WHY SASS?

A pre-processing language which provides indented syntax (its own syntax) for CSS.

Provides some features, which are used for creating stylesheets that allows writing code more efficiently and is easy to maintain.

It is a super set of CSS, which means it contains all the features of CSS and is an open source pre-processor,

Provides the document style in a good, structured format than flat CSS.

Uses re-usable methods, logic statements and some of the built-in functions such as color manipulation, mathematics and parameter lists.

SASS : VARIABLES

Think of variables as a way to store information that you want to reuse throughout your stylesheet.

You can store things like colors, font stacks, or any CSS value you think you'll want to reuse.

Sass uses the $ symbol to make something a variable.

Extremely powerful when working with brand colors and keeping them consistent throughout the site.

```
$font-stack: Helvetica, sans-serif;
$primary-color: #333;

body {
  font: 100% $font-stack;
  color: $primary-color;
}
```

SASS : NESTING

When writing HTML you've probably noticed that it has a clear nested and visual hierarchy. CSS, on the other hand, doesn't.

Sass will let you nest your CSS selectors in a way that follows the same visual hierarchy of your HTML.

Be aware that overly nested rules will result in over-qualified CSS that could prove hard to maintain and is generally considered bad practice.

```
nav {
  ul {
    margin: 0;
    padding: 0;
    list-style: none;
  }

  li { display: inline-block; }

  a {
    display: block;
    padding: 6px 12px;
    text-decoration: none;
  }
}
```

SASS : PARTIALS

You can create partial Sass files that contain little snippets of CSS that you can include in other Sass files.

A partial is a Sass file named with a leading underscore. You might name it something like _partial.scss.

The underscore lets Sass know that the file is only a partial file and that it should not be generated into a CSS file.

_partial.scss

```
nav {
  ul {
    margin: 0;
    padding: 0;
    list-style: none;
  }

  li { display: inline-block; }

  a {
    display: block;
    padding: 6px 12px;
    text-decoration: none;
  }
}
```

SASS : MIXINS

A mixin lets you make groups of CSS declarations that you want to reuse throughout your site.

It helps keep your Sass very DRY. You can even pass in values to make your mixin more flexible.

Using a file will also include the CSS it generates in your compiled output.

After you create your mixin, you can then use it as a CSS declaration starting with @include followed by the name of the mixin.

```
@mixin theme($theme: DarkGray) {
  background: $theme;
  box-shadow: 0 0 1px rgba($theme, .25);
  color: #fff;
}

.info {
  @include theme;
}
.alert {
  @include theme($theme: DarkRed);
}
.success {
  @include theme($theme: DarkGreen);
}
```

SASS : INHERITANCE

Using @extend lets you share a set of CSS properties from one selector to another.

A placeholder class is a special type of class that only prints when it is extended, and can help keep your compiled CSS neat and clean

The CSS for placeholder classes will not be generated, till it is extended.

SASS : FLOW CONTROL

Sass provides a number of at-rules that make it possible to control whether styles get emitted, or to emit them multiple times with small variations.

They can also be used in mixins and functions to write small algorithms to make writing your Sass easier.

Four Flow Control Rules:	@if controls whether or not a block is evaluated.
	@each evaluates a block for each element in a list or each pair in a map.
	@for evaluates a block a certain number of times.
	@while evaluates a block until a certain condition is met.

SASS : MODULES

Sass provides many built-in modules which contain useful functions

These modules can be loaded with the @use rule like any user-defined stylesheet, and their functions can be called like any other module member.

All built-in module URLs begin with sass: to indicate that they're part of Sass itself.

SASS : MODULES

You don't have to write all your Sass in a single file. You can split it up however you want with the @use rule.

@use rule loads another Sass file as a module, which means you can refer to its variables, mixins, and functions in your Sass file with a namespace based on the filename.

Using a file will also include the CSS it generates in your compiled output.

```scss
// _base.scss
$font-stack: Helvetica, sans-serif;
$primary-color: #333;

body {
  font: 100% $font-stack;
  color: $primary-color;
}
```

```scss
// styles.scss
@use 'base';

.inverse {
  background-color: base.$primary-color;
  color: white;
}
```

SASS : BUILT-IN MODULES

sass:math	provides functions that operate on numbers.
sass:string	makes it easy to combine, search, or split apart strings.
sass:color	making it easy to build color themes.
sass:list	lets you access and modify values in lists.
sass:map	makes it possible to look up the value associated with a key in a map
sass:selector	provides access to Sass’s powerful selector engine.
sass:meta	exposes the details of Sass’s inner workings.

REFERENCES

READING MATERIAL

- https://sass-lang.com/documentation
- https://www.tutorialspoint.com/sass/
- https://getbootstrap.com/

VIDEO TUTORIAL LINKS

- https://www.youtube.com/watch?v=_a5j7KoflTs
- https://www.youtube.com/watch?v=_kqN4hl9bGc&list=PL4cUxeGkcC9jxJX7vojNVK-o8ubDZEcNb
- https://www.youtube.com/playlist?list=PL6n9fhu94yhXd4xnk-j5FGhHjUv1LsF0V