What Is CSS Architecture?

CSS architecture produces logic to CSS authoring. Think of it as a collection of guidelines and most helpful exercises to support developers write code that is maintainable, flexible to scale, and more reusable. We accomplish that by using a modular approach, supporting the organization, and communicating meaning to codebase.

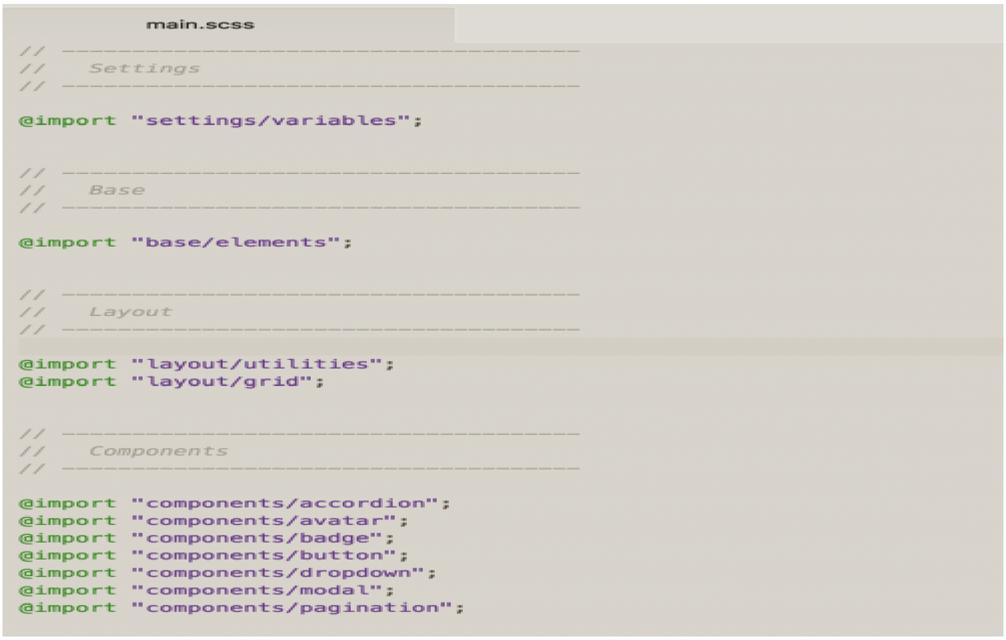
Modularity is a heart thought. Not just at a code level but more a design one. Digital products have developed a set in the earlier years, due to the growth of new devices and the responsive web. Complexity has developed. As a developers and designers should support this unusual situation. Designing pages is not the fittest way; we should make design systems. Page-based CSS won't satisfy our requirements in creating flexible and scalable products. Operating with reusable elements is the process to go.

On CSS architecture, modularity can be used on various levels. The three essential steps to beginning structuring our CSS that way are as follows

1. Breaking the code into smaller pieces and distributing them by field;
2. Coding elements in an objective and encapsulated fashion;
3. Defining CSS selectors according to their purpose and relationship with each other.

1. Separate And Categorize Your Code

To have the architecture modular, it is necessary to separate the code into smaller pieces. Many files create the code simpler to read, navigate and track. To achieve that, a CSS preprocessor – such as Sass, LESS, or Stylus – or a post-processor – such as PostCSS – is the method to continue. Preprocessors improve the abilities of CSS authoring, including innovations such as variables, mixins, and many more. To operate with separate files, your code will be broken into partials and imported on a main.scss file, which will compile everything into a single .css file.



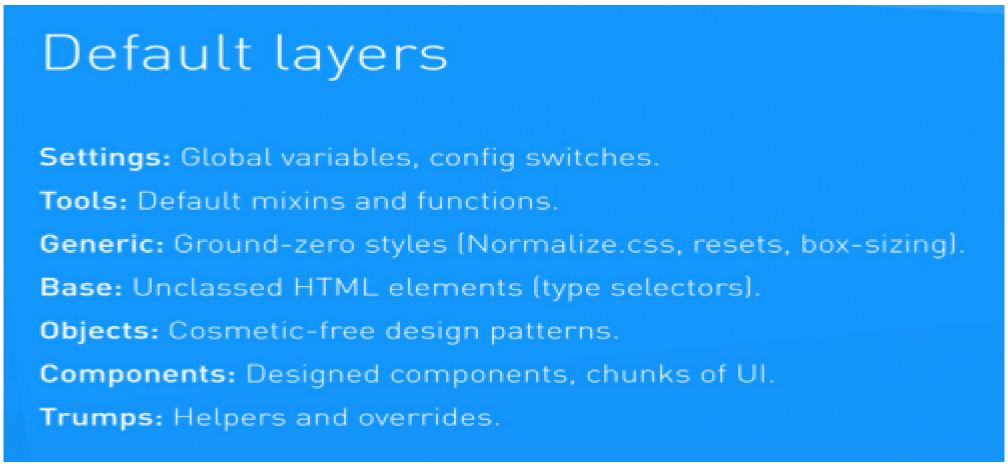
```
main.scss
// -------------------------------------------
//    Settings
// -------------------------------------------
@import "settings/variables";


// -------------------------------------------
//    Base
// -------------------------------------------
@import "base/elements";


// -------------------------------------------
//    Layout
// -------------------------------------------
@import "layout/utilities";
@import "layout/grid";


// -------------------------------------------
//    Components
// -------------------------------------------
@import "components/accordion";
@import "components/avatar";
@import "components/badge";
@import "components/button";
@import "components/dropdown";
@import "components/modal";
@import "components/pagination";
```

Presently it is time to classify each code block according to its field on the project. The first methodology to introduce selector categorization was SMACSS. The layers offered by SMACSS are **Base**, **Layout**, **Module**, **State**, and **Theme**. ITCSS extended the concept, including other layers, such as **Settings** and **Trumps**. MVCSS has a related way, but with various naming rules.



Default layers

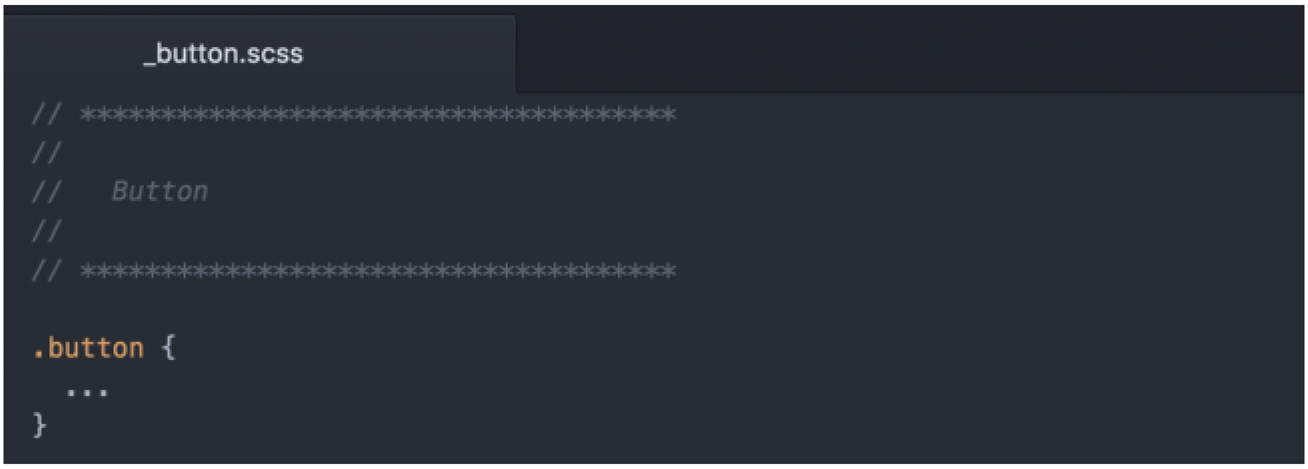
Settings: Global variables, config switches.
Tools: Default mixins and functions.
Generic: Ground-zero styles (Normalize.css, resets, box-sizing).
Base: Unclassed HTML elements (type selectors).
Objects: Cosmetic-free design patterns.
Components: Designed components, chunks of UI.
Trumps: Helpers and overrides.

2. Set your Components

Analyze components the core of a CSS architecture because that is wherever most of the code will exist. They are the reusable visible designs that make a user interface – buttons, accordions, or modals, for example.

A button component.

Every component should be encapsulated on its file. For compatibility purposes, it is good to use the related name for the file and main selector.



_button.scss

```
// ****************************************
//
//   Button
//
// ****************************************

.button {
  ...
}
```

3. Apply a naming convention

Architecture should **communicate meaning** and **support the team**. Managing a class naming convention in our components will assist us to achieve that. A pattern such as BEM (Block, Element, Modifier) will explain the purpose and relationship within a component (the Block) and its elements, as well as strengthen predictability when associated with other components. The concept behind BEM is to name things according to this structure:

- **Block**: the component itself;
- **Elements**: the essential parts of the components (Descendents on the DOM);
- **Modifiers**: changes of the block or element.

Using the blog post example:

Block

```
.blog-post { … }
```

The blog post itself.

Element

```
.blog-post__excerpt { … }
```

The excerpt inside the blog post. To denote the relation between the Element and the Block, BEM uses two underscores.

Modifier

```
.blog-post--small { … }
```

To express the relation between the Modifiers and the Block, BEM applies two dashes.

On an organizational level, this method improves **consistency** and **predictability** over the **Component** layer. On a technical level, it supports decreasing selector specificity, increasing selector efficiency, and decoupling our CSS from DOM structure (thus, **promoting reusability**).

The Goals of Great CSS Architecture

The goals of CSS architecture must not be different from the goals of all software development. CSS to be predictable, reusable, maintainable, and scalable.

Predictable

Predictable CSS indicates the guideline behaves as expected. While add or update a rule, it should not affect elements of our site that we did not expect. On little sites that infrequently change, that is not as necessary, but on large sites with tens or hundreds of pages, predictable CSS is a must.

Reusable

CSS rules should be involved and decoupled enough that can build new elements immediately from being components without producing to recode models and difficulties that you have previously solved.

Maintainable

When new elements and features require to be combined, updated, or rearranged on our site, making so should not require refactoring current CSS. Computing element X to the page should not separate element Y by its minor presence.

Scalable

As our site develops in area and complexity it regularly needs more developers to control. Scalable CSS indicates it can be efficiently handled by a single person or a large engineering team. It also determines our site's CSS architecture is simply available without needing enormous knowledge. Only because you are the only developer regarding the CSS now does not intend that will constantly be the situation.

2.2 CSS MODULES

What are CSS Modules?

CSS modules are CSS files in which all class names and animation names are scoped locally by default. Therefore CSS Modules is not an accurate spec or an implementation in the browser without a process in a build step (with the help of Webpack or Browserify) that improves class names and selectors to be scoped (i.e. kinda like namespaced).

```
<h1 class="title">An example heading</h1>
```

And that class is styled in CSS:

```
.title {
  background-color: red;
}
```

As long as that CSS is used in the HTML document, the background of that <h1> would be red. We do not require processing the CSS or the HTML. The browser understands both those file formats. CSS Modules uses a distinctive way. Instead of writing plain HTML, we require to write our entire markup in a JavaScript file, like index.js. Here's an example of how that might work.

```
import styles from "./styles.css";

element.innerHTML =
  `<h1 class="${styles.title}">
     An example heading
   </h1>`;
```

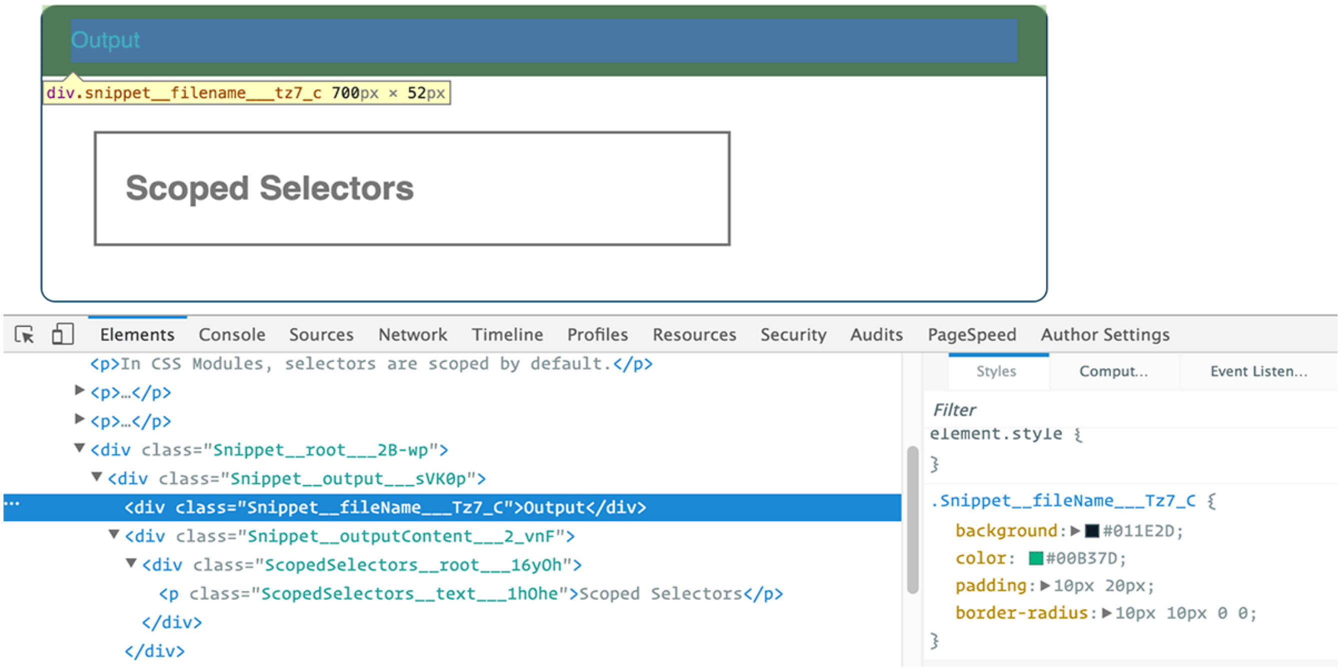
When our build step, the compiler would search for that styles.css file that we've imported, then look through the JavaScript we have written and give the .title class available via styles. title. Our build step would then process both those things into new, separate HTML and CSS files, with a new string of characters replacing both the HTML class and the CSS selector class.

Generated HTML might look like this:

```
<h1 class="_styles__title_309571057">
  An example heading
</h1>
```

Our generated CSS might look like this:

```
._styles__title_309571057 {
  background-color: red;
}
```



```
div.snippet__filename__tz7_c 700px × 52px
```

```
<p>In CSS Modules, selectors are scoped by default.</p>
▶ <p>…</p>
▶ <p>…</p>
▼ <div class="Snippet__root___2B-wp">
  ▼ <div class="Snippet__output___sVK0p">
      <div class="Snippet__fileName___Tz7_C">Output</div>
    ▼ <div class="Snippet__outputContent___2_vnF">
      ▼ <div class="ScopedSelectors__root___16yOh">
          <p class="ScopedSelectors__text___1hOhe">Scoped Selectors</p>
        </div>
      </div>
```

```
element.style {
}
.Snippet__fileName___Tz7_C {
  background: ▶ #011E2D;
  color: #00B37D;
  padding: ▶ 10px 20px;
  border-radius: ▶ 10px 10px 0 0;
}
```

The class attribute and selector .title is completely gone, replaced by this new string; our initial CSS is not being accepted to the browser at all.

The classes are dynamically created, unique, and mapped to the correct styles. This is what is meant by styles being scoped. They are scoped to particular templates. If we have a buttons.css file we would import it only into a buttons.js template and a .btn class within would be inaccessible to some other template (e.g. forms.js), unless we imported it correctly there too.

Why should we use CSS Modules?

With CSS Modules, it’s a assurance that all the styles for a single component:

1. Live in one place
2. Only apply to that component and nothing else

Plus, any component can have a true dependency, like:

```
import buttons from "./buttons.css";
import padding from "./padding.css";

element.innerHTML = `<div class="${buttons.red} ${padding.large}">`;
```

This approach is designed to fix the problem of the *global scope* in CSS.

Have you ever been influenced by a loss of time or sources to simply write CSS as soon as possible, without considering what else you might affect?
Have you ever hit some random bits and junk at the bottom of a stylesheet, meaning to get around to organizing it but never do?
Have you ever run over styles that you were not fully assured what they did or if they were even being used?
Have you ever queried if you could get relieved of some styles without breaking something? Queried if the styles held on their own or depended on other things? Or overwrote styles elsewhere?
These are issues that can point to big problems, bloated project deadlines, and sad, wistful looks out of the window.
With CSS Modules and the idea of **the local scope by default**, this problem is avoided. You are constantly required to imagine the results as you write styles.

For instance, if you do random-gross-class in HTML without applying it as a CSS modules-style class, the style will not be implemented, as the CSS selector will be transformed to ._style_random-gross-class_0038089.

The `composes` keyword

We have a module called type.css for our text styles. In that file we might have the following:

```
.serif-font {
  font-family: Georgia, serif;
}

.display {
  composes: serif-font;
  font-size: 30px;
  line-height: 35px;
}
```

We would declare one of those classes in our template like so:

```
import type from "./type.css";

element.innerHTML =
  `<h1 class="${type.display}">
    This is a heading

  </h1>`;
```

This would result in markup like:

```
<h1 class="_type__display_0980340 _type__serif_404840">
  Heading title

</h1>
```

Both classes have been bound to the element by the use of the composes keyword.

We can even compose from a specific class in a separate CSS file:

```
.element {
  composes: dark-red from "./colors.css";
  font-size: 30px;
  line-height: 1.2;

}
```

BEM not required

We do not require using BEM when we are making a CSS module. This is for two reasons:

1. **Easy parsing** – Code-like type. The show is just as clear for developers as the BEM-y .font-size__serif--large. Possibly even easier to mentally parse when the BEM selectors get long.

2. **Local scope** – We have a class like .big in one module where it adjusts the font size. In other, we handle the same class .big that enhances stuffing and font size in a different amount. It simply does not matter! They won't conflict, because the styles are scoped very carefully. Even if a module imports both stylesheets, then it has a custom name that our build method offers specifically for that class. In another word, **specificity issues leave with CSS Modules.**

2.3. CSS Framework

Why we Use CSS Framework

CSS stylesheets are difficult to keep track of, maintain, and reuse.

- Even modest style changes necessitate the creation of additional CSS rules, which may quickly turn your code into a jumbled mess.
- All CSS frameworks rely on ready-to-use classes as their fundamental building elements.
- They let you apply pre-defined stylistic rules to HTML components like margins and background colours, among other things.
- Menus, cards, and tables are all pre-built components in several frameworks.
- You can design user-friendly interfaces using these components without putting in a lot of effort.
- CSS frameworks make your styling process more efficient, clear, and easy to manage.
- You'll save time and avoid many of the difficulties that come with CSS coding if you choose one of the following frameworks.

<u>Different CSS Frameworks:-</u>

1. ***<u>BOOTSTRAP</u>***
   I can't think of a CSS framework discussion that doesn't involve Bootstrap. The framework was created by Twitter in 2011 to make responsive web design more accessible to developers. Since then, the project has grown to include support for current CSS and a plethora of features to help you increase your frontend efficiency. Bootstrap, like many other well-known products, is not without its detractors. Despite the criticism, here are several reasons why you should use it in your projects.

➢ **Reasons to use BOOTSTRAP**

- ***<u>A most prevalent frontline framework</u>***: Bootstrap is one of the most widely used open source projects. You'll always be able to find a solution to things and solve plenty of free and paid templates for practically any project.
- ***<u>Fully Featured</u>***: it's also a pre-built dynamic template with a slew of ready-to-use components. Almost everything is supported by default, from alarms to modals to navigation bars. This can make it easy for any developer to create well-structured pages, even if they have no prior frontend knowledge.
- ***<u>Mature & Supported</u>***: When the creators of smaller open source projects decide to stop working on them, they often die. Twitter introduced Bootstrap, which is now maintained by a community of hundreds of developers, ensuring reliable releases and long-term maintenance.
- ***<u>Customizable</u>***: SASS allows you to easily customise Bootstrap. You can use npm to install the project, import the bits you need, and configure practically anything with SASS variables. Learning how to use SASS to personalise Bootstrap websites will help you save a lot of time when it comes to development.

➢ **Drawbacks:**

- ***<u>It's difficult to overcome</u>***: Bootstrap has a highly distinct design and appearance that is difficult to change if you want to go for a different look.

Overriding the defaults can be difficult due to the heavy use of the important CSS rule.

- ***Overused:*** The widespread use of Bootstrap is one of the key reasons why people despise it. It has a distinct style that has become so overused among developers that the phrase "all Bootstrap websites look the same" has been coined.
- ***jQuery is used.***: Bootstrap 4 relies on jQuery for many of its interactive capabilities, unlike other CSS-only frameworks. This makes using it with JavaScript frameworks like React orVue more challenging, but not impossible. Fortunately, the jQuery reliance will be removed in Boostrap 5, which will be released soon.
- ***It's important to include:*** All of Bootstrap's advantages come at a cost – it's rather cumbersome to use in your projects. It is not as lightweight or modular as the other frameworks listed here, despite the fact that you can import parts of the project.

2. *Foundation:*

Foundation is ideal for experienced developers who desire the independence of a framework but also want the strength of a full-featured framework. In actuality, Foundation is a collection of frontend development tools, not just a CSS framework. These tools can be used in conjunction with one another or on their own. Foundation for Sites provides the foundation for building web pages, while Foundation for Emails allows you to create visually appealing emails that can be viewed on any device. The final element of the puzzle is Motion UI, which allows you to design complex CSS animations. ZURB, the firm behind numerous open source Javascript and CSS projects, develops and maintains Foundation. This framework has taken a lot of care to build, and ZURB uses it extensively in their own projects.

➢ Reasons to use Foundation:

- ***Style that is generic***: Foundation, unlike Bootstrap, does not have a specific style for its components. Its modular and flexible components come in a variety of styles and may be easily altered.
- ***Fully Featured***: Foundation comes with practically all of the components you'll need. A developer-friendly grid system, navigation bars, and different container types are all included. Foundation also gives you access to pre-made HTML templates, either generated by the development team or by the community, that you may use to get started on projects that are tailored to your specific requirements.
- ***Email Design:*** Foundation comes with practically all of the components you'll need. A developer-friendly grid system, navigation bars, and different container types are all included. Foundation also gives you access to pre-made HTML templates, either generated by the development team or by the

community, that you may use to get started on projects that are tailored to your specific requirements.

- ***Animations:*** Foundation works well with ZURB's Motion UI framework, which enables you use built-in effects to create transitions and animations. Your ideas will come to life when you use Motion UI in conjunction with Foundation!

➢ **Drawbacks:**

- Hard to learn: There are almost too many options when it comes to foundation. It contains a lot of features and is a lot more complicated than other frameworks. When creating frontend layouts, it allows you a lot of flexibility, but you must first grasp how everything works.
- Relies on JavaScript: There are almost too many options when it comes to foundation. It contains a lot of features and is a lot more complicated than other frameworks. When creating frontend layouts, it allows you a lot of flexibility, but you must first grasp how everything works.

3. ***Bulma:***

Bulma is a wonderful alternative to Bootstrap because it has a distinctive appearance and modern code. It's simple to use and integrate into your projects, and it includes a variety of pre-made components. It has received a lot of appreciation for its concise syntax and basic yet appealing look. It's a foundation that can turn a drab website into something vibrant and engaging. It's no longer a niche framework, with over 40,000 stars on GitHub, but rather a force to be reckoned with.

➢ **Reasons to use Bulma:**

- ***Aesthetic design***: Bulma, in my opinion, is the most attractive CSS framework on this list. It has a clean and modern style, so even if you leave the defaults alone, you'll have a great-looking webpage.
- ***Modern***: Technology evolves, and what was once complicated may suddenly be easy. Bulma was one of the first flexbox-based frameworks to apply the new concepts, thanks to CSS's flexbox layout module, which made it easier to develop responsive layouts.
- **Developer Friendly**: Bulma's creators strive to deliver a wonderful experience to the developer, while frontend developers want to deliver a wonderful experience to the end-user. Bulma was designed with easy-to-use and remember naming standards in mind.
- ***Easy to customize***: SASS can be used to change Bulma's colours, paddings, and many other default properties. You may quickly set up the settings for your project this way.
- ***No JavaScript***: JavaScript functionalities are not available in Bulma. It's easy to combine with JavaScript frameworks like Vue or React because it's solely CSS-based.

➢ **Drawbacks:**

- ***Distinct Style***: Bulma's distinct style has both positive and negative aspects. Because it is so different, if it is overdone, we can end up with websites that seem extremely identical, as is the situation with Bootstrap.
- ***Less Complete***: Bulma competes with Bootstrap in many ways, but when it comes to accessibility and other enterprise-grade capabilities, it falls short.

4. ***Tailwind CSS:***

Most CSS frameworks try to do too much" — Tailwind's tagline explains why it is a lightweight framework that gives developers a lot of options. It does not come with a pre-designed layout, but rather allows you to quickly adopt your own personal style. It achieves this by providing utility classes that almost eliminate the need for CSS coding. Its tremendous capabilities entice experienced frontend developers, who use it throughout their projects.

➢ Reasons to use Tailwind CSS:

- ***Atomic CSS***: You would ordinarily code in CSS to centre an element, create a flexible layout, or use a specified text colour. Tailwind's powerful utility classes make all of these common styles simple to create. An HTML element's classes explicitly specify what it will look like, which is frequently referred to as Atomic CSS. For instance, <div class="m-1 text-centerbg-black">...</div> will display an element with a margin of 1 (i.e. small margin), centered text, and a black background.
- ***No design***: Tailwind does not include any pre-built components or a design language. This means you won't have to overwrite current styles, and you'll be able to create custom designs faster.
- ***Reusable Components***: Despite the fact that Tailwind does not come with any pre-designed components, it does allow you to construct your own custom components that you may reuse across several projects. On the official website, you may also find several component samples to utilise as a starting point.
- ***SASS Integration***: Install and import Tailwind into your SASS or PostCSS project to get the most out of it. This enables you to take advantage of all of Tailwind's features in order to create more effective CSS. The @apply syntax "copies" Tailwind rules into your SASS or CSS code, allowing you to continue authoring CSS while gaining superpowers!

➢ Drawbacks

- ***Steep Learning Curve***: For less experienced developers, Tailwind is not the ideal option. You must thoroughly understand how frontend technologies work because it does not give pre-made components. Tailwind has a steep learning curve because you must master the syntax in order to use the framework effectively.

- ***No directly used***: Tailwind, like other frameworks, may be applied to your project as a packaged CSS file. However, according to the official installation instruction, doing so will disable many of the framework's features and prevent you from using the compressed version (27 KB compressed vs 348 KB raw). You'll need to know how to utilise Webpack, Gulp, or another frontend framework to get the most out of Tailwind.

5. ***UIKit:***

UIKit is a frontend framework that lets you import only the functionality you require. It has over 16k stars on GitHub, and developers choose it because of its simple API and clean design. A pro edition of UIKit is also available, which includes themed pages for WordPress and Joomla, as well as an easy-to-use page builder.

➢ Reasons to use UIKit

- ***Dozens of Components***: UIKit has a large number of components that allow you to create complicated frontend layouts. It comes with all of the standard utilities and components, but it also contains sophisticated elements like navigation bars, off-canvas sidebars, and parallax designs.
- ***Extensible***: Using the LESS or SASS preprocessors, UIKit may be easily tweaked and enhanced.
- ***UI-Based Customizer***: UIKit has a web-based customizer that allows you to make changes to the design in real time before copying the SASS or LESS variables into your project. This feature of UIKit can feel like magic and help you get started on new projects quickly.

➢ Drawbacks:

- ***Complex for Smaller Projects***: Because UIKit is a complicated framework that demands a thorough understanding of frontend programming, it is not recommended for inexperienced developers. It's ideal for complex applications, but it may be overkill for smaller ones.
- ***Smaller Community***: Despite the fact that its npm package is downloaded 27,000 times each week, it is not as well-known as other frameworks. It will be more difficult to find solutions or hire employees with UIKit experience than it will be with Bootstrap or Foundation.

BOOTSTRAP 4 INTRODUCTION

Using the MLA Style ensured that everyone who followed the same format, even if they were thousands of miles apart or written five years later, would have the same results. For online design, author believes Bootstrap works similarly. Regardless of where you live or who you work for, it helps developers work more efficiently and write CSS in a clean and consistent manner. It also ensures that your website follows a mobile-first approach and is compatible with a variety of browsers and devices.Bootstrap, in their opinion, will substantially improve the way you develop websites and save you a lot of time.

DEFINITION OF BOOTSTRAP

*“**Bootstrap 4 is a popular and effective mobile-first front-end framework for creating responsive mobile-first websites. It's the most recent version of Bootstrap, and it works with HTML, CSS, and JavaScript.**”*

Purpose of BOOTSTRAP 4

- Instead of employing mobile first styles in individual files, it uses them across the library.
- Provides a simple and consistent way for creating a developer interface.
- It comes with a number of attractive and effective built-in components that are simple to personalise.
- It's free and open source, with web-based customization.
- Anyone with a basic understanding of HTML and CSS can get started with Bootstrap. The official Bootstrap site also provides excellent documentation.
- All major browsers support it, and its responsive CSS adjusts to Desktops, Tablets, and Mobiles.

Environment Setup

You may use Bootstrap 4 in your website by including it through a CDN or downloading it from **(getbootstrap.com.)**By incorporating it from the Content Delivery Network, Bootstrap 4 can be used in the website. In your project, use the Bootstrap CDN of CSS and JS that has been prepared below.

```
<!-- Compiled and Minified Bootstrap CSS -->
<linkrel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css"
integrity="sha384-
Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm"
crossorigin="anonymous">

<!--jQuery Library -->
<scriptsrc="https://code.jquery.com/jquery-3.2.1.slim.min.js"
integrity="sha384-
KJ3o2DKtIkvYIK3UENzmM7KCkRr/rE9/Qpg6aAZGJwFDMVNA/GpGFF93hXpG5KkN"
crossorigin="anonymous">
</script>
```

```
<!-- Popper -->
<scriptsrc="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.12.9/umd/popper.min.js"
integrity="sha384-
ApNbgh9B+Y1QKtv3Rn7W3mgPxhU9K/ScQsAP7hUibX39j7fakFPskvXusvfa0b4Q"
crossorigin="anonymous">
</script>

<!-- Compiled and Minified Bootstrap JavaScript -->
<scriptsrc="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstrap.min.js"
integrity="sha384-
JZR6Spejh4U02d8jOt6vLEHfe/JQGiRRSQQxSfFWpi1MquVdAyjUar5+76PVCmYl"
crossorigin="anonymous">
</script>
```

[Ref:-https://www.tutorialspoint.com/bootstrap4/bootstrap4_environment_setup.html]

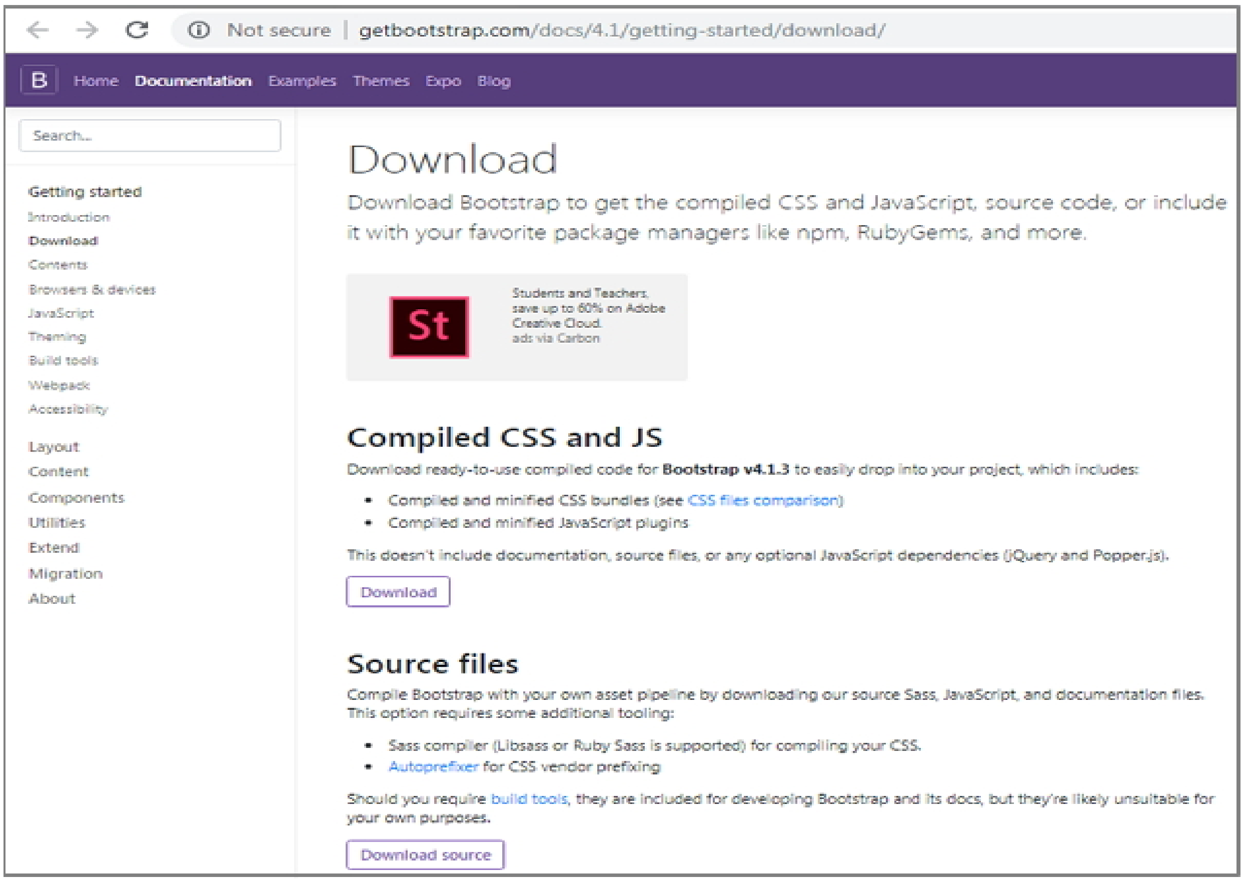
Downloading BOOTSTRAP4:

You can download the Bootstrap 4 from **https://getbootstrap.com/docs/4.1/getting-started/download/.** When you click on this link, you will get to see a screen as shown below –

Here you can see two buttons −

- **Download** − Clicking this, you can download the precompiled and minified versions of Bootstrap's CSS and JavaScript. No documentation or original source code files are included.
- **Download Source** − Clicking this, you can get the latest Bootstrap SCSS, JavaScript source code and documentation files.

For better understanding and ease of use, we shall use precompiled version of Bootstrap throughout the tutorial. As the files are complied and minified, you don't have to bother every time including separate files for individual functionality.



Download

Download Bootstrap to get the compiled CSS and JavaScript, source code, or include it with your favorite package managers like npm, RubyGems, and more.

Compiled CSS and JS

Download ready-to-use compiled code for **Bootstrap v4.1.3** to easily drop into your project, which includes:

- Compiled and minified CSS bundles (see CSS files comparison)
- Compiled and minified JavaScript plugins

This doesn't include documentation, source files, or any optional JavaScript dependencies (jQuery and Popper.js).

Source files

Compile Bootstrap with your own asset pipeline by downloading our source Sass, JavaScript, and documentation files. This option requires some additional tooling:

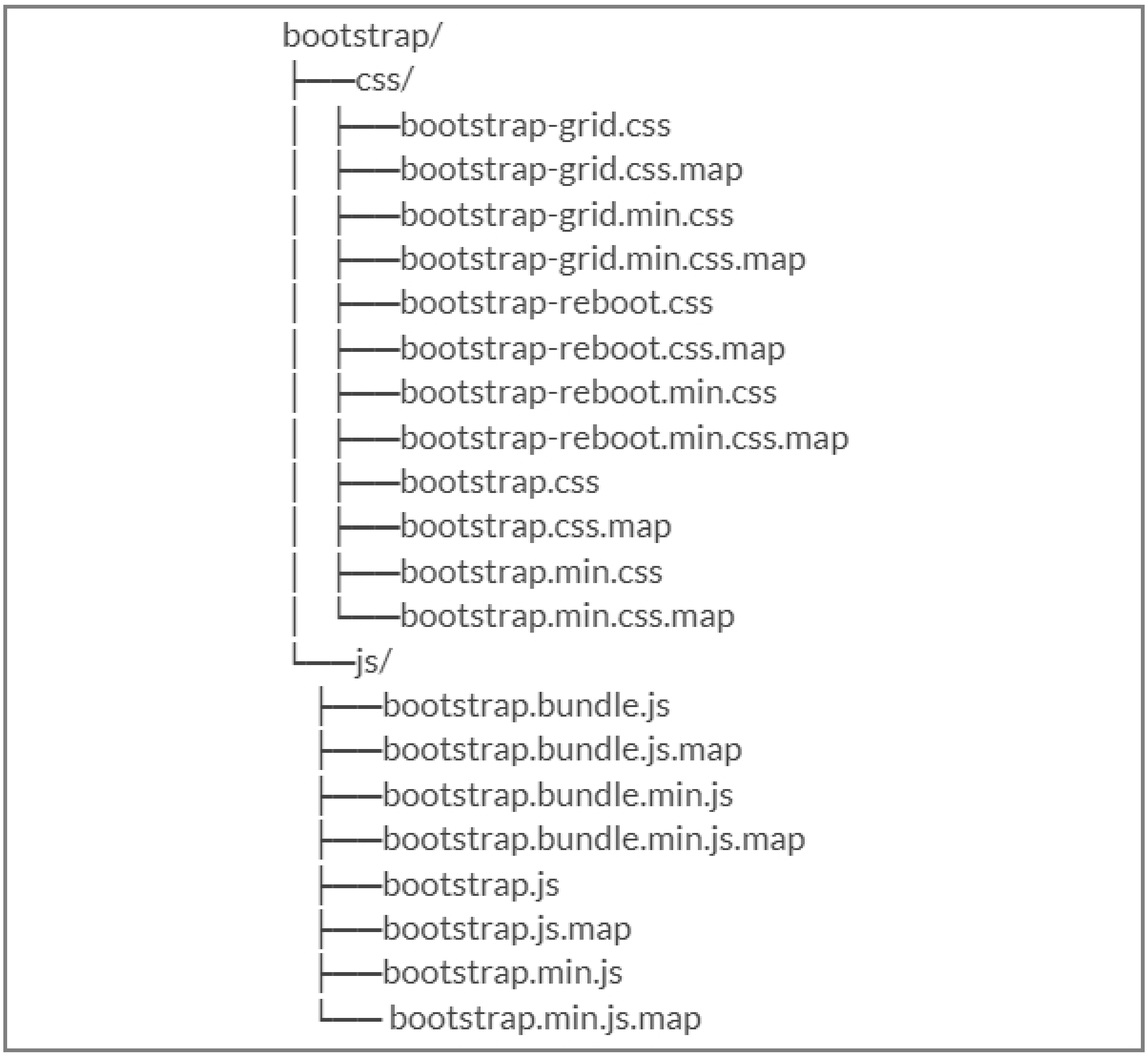
- Sass compiler (Libsass or Ruby Sass is supported) for compiling your CSS.
- Autoprefixer for CSS vendor prefixing

Should you require build tools, they are included for developing Bootstrap and its docs, but they're likely unsuitable for your own purposes.

File Structure

Precompiled Bootstrap 4

Once the compiled version Bootstrap 4 is downloaded, extract the ZIP file, and you will see the following file/directory structure −



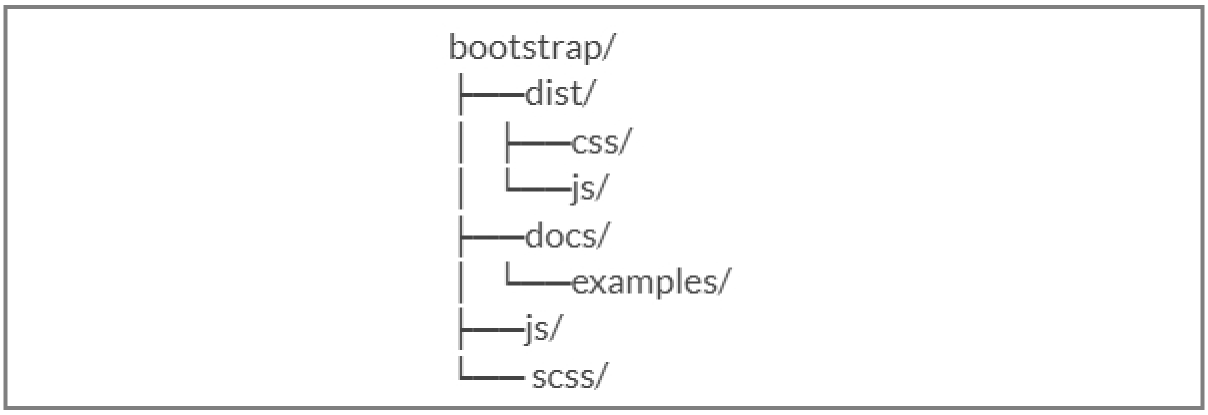
```
bootstrap/
├──css/
│   ├──bootstrap-grid.css
│   ├──bootstrap-grid.css.map
│   ├──bootstrap-grid.min.css
│   ├──bootstrap-grid.min.css.map
│   ├──bootstrap-reboot.css
│   ├──bootstrap-reboot.css.map
│   ├──bootstrap-reboot.min.css
│   ├──bootstrap-reboot.min.css.map
│   ├──bootstrap.css
│   ├──bootstrap.css.map
│   ├──bootstrap.min.css
│   └──bootstrap.min.css.map
└──js/
   ├──bootstrap.bundle.js
   ├──bootstrap.bundle.js.map
   ├──bootstrap.bundle.min.js
   ├──bootstrap.bundle.min.js.map
   ├──bootstrap.js
   ├──bootstrap.js.map
   ├──bootstrap.min.js
   └── bootstrap.min.js.map
```

[Ref:-Wikipedia, [1]]

As you can see, there are compiled CSS and JS (bootstrap.*), as well as compiled and minified CSS and JS (bootstrap.min.*).

Bootstrap 4 Source Code

If you have downloaded the Bootstrap 4 source code, then the file structure would be as follows −



```
bootstrap/
├──dist/
│  ├──css/
│  └──js/
├──docs/
│  └──examples/
├──js/
└── scss/
```

[ref:-wikipedia]

- The files under *js/* and *scss/* are the source code for Bootstrap CSS and JavaScript.
- The *dist/* folder include everything listed in the precompiled download section above.
- The *docs/examples/*, includes source code for Bootstrap documentation and examples of Bootstrap usage.

Bootstrap 4 uses container classes to wrap the page's contents. It contains two container classes −

- **.container** − It represents a fixed width container.
- **.container-fluid** − It represents a full width container.

a) Container

The *.container* class is used to wrap the page content with fixed width and content can be placed in the center easily by using the *.container* class as shown below.

SYNTAX

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

b) Fluid Container

You can create a full width container by using the *.container-fluid* class as shown below.

SYNTAX

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

Grid System

Bootstrap 4 grid system with flexbox is fully responsive and scalable up to 12 columns (depending on device size) by building a web layout with rows and columns. It has a mobile-first responsive fluid grid structure that scales the columns as the device or viewport size grows.

Working of Grid System

- For appropriate alignment and padding, rows must be placed within a.container class.
- Use the.container class for responsive width and the.container-fluid class for fixed width across all viewports.
- Create horizontal groups of columns by using rows.
- Only columns may be the direct children of rows, and content should be placed within the columns.
- Padding is used to adjust the amount of space between columns.
- If you have more than 12 columns in a row, they will be separated by a new line.
- Padding between columns creates gaps in the content. With the.no-gutters class on the row, you may remove the margin from rows and the padding from columns.
- Using five grid breakpoints, such as extra small, small, medium, large, and extra big, you may make a grid system responsive.
- For fast creating grid layouts, predefined grid classes such as.col-4 are available.LESS mixins can also be used for more semantic layouts.

Grid Options

The following table summarizes aspects of how Bootstrap 4 grid system works across multiple devices −

	Extra small devices (<576px)	**Small devices (≥576px)**	**Medium devices (≥768px)**	**Large devices (≥992px)**	**Extra Large devices (≥1200px)**
Grid behavior	Horizontal at all times	Collapsed to start, horizontal above breakpoints	Collapsed to start, horizontal above breakpoints	Collapsed to start, horizontal above breakpoints	Collapsed to start, horizontal above breakpoints
Max container width	None (auto)	540px	720px	960px	1140px
Class classes	**.col-**	**.col-sm-**	**.col-md-**	**.col-lg-**	**.col-xl-**

# of columns	12	12	12	12	12
Gutter width	30px (15px on each side of a column)	30px (15px on each side of a column)	30px (15px on each side of a column)	30px (15px on each side of a column)	30px (15px on each side of a column)
Nestable	Yes	Yes	Yes	Yes	Yes
Column ordering	Yes	Yes	Yes	Yes	Yes

[Ref:-https://www.tutorialspoint.com/bootstrap4/bootstrap4_environment_setup.html]

Basic Grid Structure

Following is basic structure of Bootstrap 4 grid −

```
<divclass="container">
<divclass="row">
<divclass="col-*-*"></div>
<divclass="col-*-*"></div>
</div>

<divclass="row">
<divclass="col-*-*"></div>
<divclass="col-*-*"></div>
<divclass="col-*-*"></div>
</div>

<divclass="row">...</div>
</div>
```

Content

For displaying text, code blocks, responsive pictures, data in a tabular format, and other content on a web page, Bootstrap 4 employs a number of content methods.

The content methods that you can use to edit the Bootstrap 4 are listed in the table below.

S.No.	Methods & Description
1	Typography Headings, paragraphs, lists, and other inline components are created using the typography feature.
2	Code It's used in the document to show inline and multiline code chunks.
3	Images The <img> tag in Bootstrap 4 is used to support images.
4	Tables Tables are used to provide information in a tabular style.
5	Figures The text, as well as accompanying images and an optional caption, is specified in the figure element.

[ref:-1,2,3, Wikipedia, 4]

Components

For displaying text, code blocks, responsive pictures, data in a tabular format and other content on a web page, Bootstrap 4 employs a number of content methods.

The content methods that you can use to edit the Bootstrap 4 are listed in the table below.

S.No.	Methods & Description
1	Alerts The predefined message for user activities is specified by the alert component.

2	Badges The predefined message for user activities is specified by the alert component.
3	Breadcrumb It's used to display a site's hierarchy-based information.
4	Buttons Clickable buttons are provided by Bootstrap for displaying content such as text and graphics.
5	Button group Multiple buttons can be placed together on a single line using button groups.
6	Cards A card is a type of content container that displays a box with a border and padding around it.
7	Carousel Carousel is a responsive and versatile way to add a slider to your website.
8	Collapse It's used to reveal or conceal content.
9	Dropdowns Links can be shown in a list fashion using dropdown menus.
10	Forms The form element is used to gather user input.
11	Input group You may quickly prepend and append text or buttons to text-based inputs by using input groups.
12	Jumbotron It enlarges headlines and provides a lot of space for landing page text.

13	Modal A modal window is a sub-window that is stacked on top of its parent window.
14	Navs In a horizontal menu, Bootstrap provides navigation items for your site.
15	Navbar For your application or website, Navbar provides navigation headers.
16	Pagination Pagination is a technique for dividing content into numerous pages.
17	Popovers Popover is similar to a tooltip in that it provides an expanded view with a heading.
18	Progress With stacked bars, dynamic backgrounds, and text labels, a progress bar depicts the progress of a procedure.
19	Scrollspy Scrollspy is used to show which link in the menu is now active dependent on the scroll position.
20	Tooltips When describing a link, tooltips come in handy.

[ref:- 1, 6, 4, 2]

Utilities

On the web page, Bootstrap 4 employs a suite of utilities for displaying borders, text colour, and video embedding, among other things.

The utilities types that you can use to manipulate the Bootstrap 4 are listed in the table below.

S.No.	Methods & Description
1	Borders The Border utility allows you to customise the style, colour, and radius of an element's border.
2	Clearfix and Close Icon Clearfix is used to clear the floated content, while the close icon is used to dismiss it.
3	Colors To modify the colour of text, links, and the background colour of an element, use contextual classes.
4	Embed The <iframe> element is used to embed the video in a page.
5	Float It's used to make an element float to the left or right.
6	Shadows and Spacing The shadow tool adds a shadow to an element, whereas the spacing utility gives it margin or padding values.
7	Sizing The width and height utilities can be used to make an element wider or taller.
8	Text You can use Bootstrap's text utilities to change text alignment, morph, weight, and more.
9	Flex The Flex utility can be used to control the page's layout, alignment, grid columns, navigation, and other elements.

2.4. Selectors and Pseudo Classes

2.4. Selectors and Pseudo Classes

CSS Selectors

A CSS selector selects the HTML element(s) you want to style.CSS selectors are used to "find" (or select) the HTML elements you want to style.

The style rules associated with that selector will be applied to the elements that match the selector pattern.

Selectors are one of the most important aspects of CSS as they allow you to target specific elements on your web page in various ways so that they can be styled.

We can divide CSS selectors into five categories:

- **Simple selectors** (select elements based on name, id, class)
- **Combinator selectors** (select elements based on a specific relationship between them)
- **Pseudo-class selectors** (select elements based on a certain state)
- **Pseudo-elements selectors** (select and style a part of an element)
- **Attribute selectors** (select elements based on an attribute or attribute value)

Selector	**Example**	**Example description**
.class	.intro	Selects all elements with class="intro"
.class1.class2	.name1.name2	Selects all elements with both *name1* and *name2* set within its class attribute
.class1 .class2	.name1 .name2	Selects all elements with *name2* that is a descendant of an element with *name1*
#id	#firstname	Selects the element with id="firstname"
*	*	Selects all elements
element	p	Selects all <p> elements
element.class	p.intro	Selects all <p> elements with class="intro"
element,element	div, p	Selects all <div> elements and all <p> elements
element element	div p	Selects all <p> elements inside <div> elements
element>element	div > p	Selects all <p> elements where the parent is a <div> element
element+element	div + p	Selects the first <p> element that is placed immediately after <div>

		elements
element1~element2	p ~ ul	Selects every <ul> element that is preceded by a <p> element
[*attribute*]	[target]	Selects all elements with a target attribute
[*attribute=value*]	[target=_blank]	Selects all elements with target="_blank"
[*attribute~=value*]	[title~=flower]	Selects all elements with a title attribute containing the word "flower"
[*attribute|=value*]	[lang|=en]	Selects all elements with a lang attribute value starting with "en"
[*attribute^=value*]	a[href^="https"]	Selects every <a> element whose href attribute value begins with "https"
[*attribute$=value*]	a[href$=".pdf"]	Selects every <a> element whose href attribute value ends with ".pdf"
[*attribute*=value*]	a[href*="w3schools"]	Selects every <a> element whose href attribute value contains the substring "w3schools"
:active	a:active	Selects the active link
::after	p::after	Insert something after the content of each <p> element
::before	p::before	Insert something before the content of each <p> element
:checked	input:checked	Selects every checked <input> element
:default	input:default	Selects the default <input> element
:disabled	input:disabled	Selects every disabled <input> element
:empty	p:empty	Selects every <p> element that has no children (including text nodes)
:enabled	input:enabled	Selects every enabled <input> element
:first-child	p:first-child	Selects every <p> element that is the first child of its parent
::first-letter	p::first-letter	Selects the first letter of every <p> element
::first-line	p::first-line	Selects the first line of every <p> element

:first-of-type	p:first-of-type	Selects every <p> element that is the first <p> element of its parent
:focus	input:focus	Selects the input element which has focus
:fullscreen	:fullscreen	Selects the element that is in full-screen mode
:hover	a:hover	Selects links on mouse over
:in-range	input:in-range	Selects input elements with a value within a specified range
:indeterminate	input:indeterminate	Selects input elements that are in an indeterminate state
:invalid	input:invalid	Selects all input elements with an invalid value
:lang(*language*)	p:lang(it)	Selects every <p> element with a lang attribute equal to "it" (Italian)
:last-child	p:last-child	Selects every <p> element that is the last child of its parent
:last-of-type	p:last-of-type	Selects every <p> element that is the last <p> element of its parent
:link	a:link	Selects all unvisited links
::marker	::marker	Selects the markers of list items
:not(*selector*)	:not(p)	Selects every element that is not a <p> element
:nth-child(*n*)	p:nth-child(2)	Selects every <p> element that is the second child of its parent
:nth-last-child(*n*)	p:nth-last-child(2)	Selects every <p> element that is the second child of its parent, counting from the last child
:nth-last-of-type(*n*)	p:nth-last-of-type(2)	Selects every <p> element that is the second <p> element of its parent, counting from the last child
:nth-of-type(*n*)	p:nth-of-type(2)	Selects every <p> element that is the second <p> element of its parent
:only-of-type	p:only-of-type	Selects every <p> element that is the only <p> element of its parent
:only-child	p:only-child	Selects every <p> element that is the

		only child of its parent
:optional	input:optional	Selects input elements with no "required" attribute
:out-of-range	input:out-of-range	Selects input elements with a value outside a specified range
::placeholder	input::placeholder	Selects input elements with the "placeholder" attribute specified
:read-only	input:read-only	Selects input elements with the "readonly" attribute specified
:read-write	input:read-write	Selects input elements with the "readonly" attribute NOT specified
:required	input:required	Selects input elements with the "required" attribute specified
:root	:root	Selects the document's root element
::selection	::selection	Selects the portion of an element that is selected by a user
:target	#news:target	Selects the current active #news element (clicked on a URL containing that anchor name)
:valid	input:valid	Selects all input elements with a valid value
:visited	a:visited	Selects all visited links

CSS element Selector

The element selector selects HTML elements based on the element name. An element type selector matches all instance of the element in the document with the corresponding element type name.

Example:

```
<!DOCTYPE html>
<html>
<head>
<style>
p {
  text-align: center;
  color: green;
}
```

```
</style>
</head>
<body>
<p>The standard Lorem Ipsum passage, used since the 1500s</p>
<p id="para1">Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod
        tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis
        nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.</p>
<p> Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat
        nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia
        deserunt mollit anim id est laborum.</p>
</body>
</html>
```

CSS id Selector

The id selector uses the id attribute of an HTML element to select a specific element. The id of an element is unique within a page, so the id selector is used to select one unique element. To select an element with a specific id, write a hash (#) character, followed by the id of the element.

Example:

```
<!DOCTYPE html>
<html>
<head>
<style>
#para1 {
  text-align: center;
  color: red;
}
</style>
</head>
<body>
<p id="para1">The standard Lorem Ipsum passage, used since the 1500s</p>
<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor
incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud
exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.</p>
</body>
</html>
```

CSS class Selector

The class selector selects HTML elements with a specific class attribute.

To select elements with a specific class, write a period (.) character, followed by the class name.

Example

```
<!DOCTYPE html>
<html>
<head>
<style>
.center {
  text-align: center;
  color: red;
}
</style>
</head>
<body>
<h1 class="center">The standard Lorem Ipsum passage, used since the 1500s</h1>
<p class="center">Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do
    eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim
    veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo
    consequat.</p>
</body>
</html>
```

You can also specify that only specific HTML elements should be affected by a class.

Example

In this example only <p> elements with class="center" will be red and center-aligned:

```
p.center {
  text-align: center;
  color: red;
}
```

CSS Universal Selector

The universal selector (*) selects all HTML elements on the page.

The universal selector may be omitted if other conditions exist on the element. This selector is often used to remove the default margins and paddings from the elements for quick testing purpose.

Example:

```
<!DOCTYPE html>
<html>
<head>
<style>
* {
  text-align: center;
  color: red;
}
</style>
</head>
<body>
<h1>The standard Lorem Ipsum passage, used since the 1500s</h1>
<p id="para1">Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod
        tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis
        nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.</p>
<p> Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla
        pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia
        deserunt mollit anim id est laborum.</p>
</body>
</html>
```

CSS Grouping Selector

The grouping selector selects all the HTML elements with the same style definitions.

Example:

```
<!DOCTYPE html>
<html>
<head>
<style>
h1, h2, p {
  text-align: center;
  color: blue;
}
</style>
</head>
<body>
<h1>The standard Lorem Ipsum passage, used since the 1500s</h1>
<h2>1914 translation by H. Rackham</h2>
<p id="para1">Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do
eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam,
quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.</p>
</body>
```

```
        </html>
```

Pseudo Classes

A pseudo-class is used to define a special state of an element.

For example, it can be used to:

- Style an element when a user mouses over it
- Style visited and unvisited links differently
- Style an element when it gets focus

Syntax:

The syntax of pseudo-classes are:

```
selector:pseudo-class {
  property: value;
}
```

All CSS Pseudo Classes

Selector	Example	Example description
:active	a:active	Selects the active link
:checked	input:checked	Selects every checked <input> element
:disabled	input:disabled	Selects every disabled <input> element
:empty	p:empty	Selects every <p> element that has no children
:enabled	input:enabled	Selects every enabled <input> element
:first-child	p:first-child	Selects every <p> elements that is the first child of its parent
:first-of-type	p:first-of-type	Selects every <p> element that is the first <p> element of its parent
:focus	input:focus	Selects the <input> element that has focus
:hover	a:hover	Selects links on mouse over
:in-range	input:in-range	Selects <input> elements with a value within a specified range
:invalid	input:invalid	Selects all <input> elements with an invalid value
:lang(*language*)	p:lang(it)	Selects every <p> element with a

		lang attribute value starting with "it"
:last-child	p:last-child	Selects every <p> elements that is the last child of its parent
:last-of-type	p:last-of-type	Selects every <p> element that is the last <p> element of its parent
:link	a:link	Selects all unvisited links
:not(selector)	:not(p)	Selects every element that is not a <p> element
:nth-child(n)	p:nth-child(2)	Selects every <p> element that is the second child of its parent
:nth-last-child(n)	p:nth-last-child(2)	Selects every <p> element that is the second child of its parent, counting from the last child
:nth-last-of-type(n)	p:nth-last-of-type(2)	Selects every <p> element that is the second <p> element of its parent, counting from the last child
:nth-of-type(n)	p:nth-of-type(2)	Selects every <p> element that is the second <p> element of its parent
:only-of-type	p:only-of-type	Selects every <p> element that is the only <p> element of its parent
:only-child	p:only-child	Selects every <p> element that is the only child of its parent
:optional	input:optional	Selects <input> elements with no "required" attribute
:out-of-range	input:out-of-range	Selects <input> elements with a value outside a specified range
:read-only	input:read-only	Selects <input> elements with a "readonly" attribute specified
:read-write	input:read-write	Selects <input> elements with no "readonly" attribute
:required	input:required	Selects <input> elements with a "required" attribute specified
:root	Root	Selects the document's root element
:target	#news:target	Selects the current active #news element (clicked on a URL containing that anchor name)

:valid	input:valid	Selects all <input> elements with a valid value
:visited	a:visited	Selects all visited links

All CSS Pseudo Elements

Selector	**Example**	**Example description**
::after	p::after	Insert content after every <p> element
::before	p::before	Insert content before every <p> element
::first-letter	p::first-letter	Selects the first letter of every <p> element
::first-line	p::first-line	Selects the first line of every <p> element
::selection	p::selection	Selects the portion of an element that is selected by a user

Pseudo-classes and CSS Classes

Pseudo-classes can be combined with CSS classes.

When you hover over the link in the example, it will change color of text:

Example:

```
<!DOCTYPE html>
<html>
<head>
<style>
a.highlight:hover {
  color: #ff0000;
}
</style>
</head>
<body>
<h2>Pseudo-classes and CSS Classes</h2>
<p>When you hover over the first link below, it will change color:</p>
<p><a class="highlight" href="#">CSS Syntax</a></p>
<p><a href="#">CSS Tutorial</a></p>
</body>
</html>
```

Example: An example of using the `:hover` pseudo-class on a <div> element

```
<!DOCTYPE html>
<html>
<head>
<style>
div {
  background-color: green;
  color: white;
  padding: 25px;
  text-align: center;
}
div:hover {
  background-color: blue;
}
</style>
</head>
<body>
<p>Mouse over the div element below to change its background color:</p>

<div>Mouse Over Me</div>
</body>
</html>
```

The :first-child Pseudo-class

The `:first-child` pseudo-class matches a specified element that is the first child of another element.

Example:

```
<!DOCTYPE html>
<html>
<head>
<style>
p:first-child {
  color: red;
}
</style>
</head>
<body>
<p> Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor
    incididunt ut labore et dolore magna aliqua.</p>
<p> Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea
    commodo consequat.</p>
</body>
</html>
```

Example: the selector matches the first <i> element in all <p> elements

```
<!DOCTYPE html>
<html>
<head>
<style>
p i:first-child {
  color: blue;
}
</style>
</head>
<body>
<p>I am a <i>strong</i> person. I am a <i>strong</i> person.</p>
<p>I am a <i>strong</i> person. I am a <i>strong</i> person.</p>
</body>
</html>
```

2.5. Fonts and Text Effects

Fonts

Choosing the right font has a huge impact on how the readers experience a website. The right font can create a strong identity for your brand. Using a font that is easy to read is important. The font adds value to your text. It is also important to choose the correct color and text size for the font.

You can set following font properties to an element −

- The **font-family** property is used to change the face of a font.
- The **font-style** property is used to make a font italic.
- The **font-variant** property is used to create a small-caps effect.
- The **font-weight** property is used to increase or decrease how bold or light a font appears.
- The **font-size** property is used to increase or decrease the size of a font.
- The **font** property is used as shorthand to specify a number of other font properties.

Set the Font Family

Following is the example, which demonstrates how to set the font family of an element. Possible value could be any font family name.

```
<!DOCTYPE html>
<html>
  <head>
  </head>

  <body>
    <p style = "font-family:Times New Roman;"> Contrary to popular belief, Lorem Ipsum is
not simply random text. </p>
  </body>
</html>
```

Set the Font Style

Following is the example, which demonstrates how to set the font style of an element. Possible values are *normal, italic and oblique*.

```
<!DOCTYPE html>
<html>
  <head>
  </head>
  <body>
```

```
    <p style = "font-style:italic;">  Contrary to popular belief, Lorem Ipsum is not simply
random text. </p>
  </body>
</html>
```

Set the Font Variant

The following example demonstrates how to set the font variant of an element. Possible values are *normal and small-caps*.

```
<!DOCTYPE html>
<html>
  <head>
  </head>

  <body>
    <p style = "font-variant:small-caps;"> Contrary to popular belief, Lorem Ipsum is not
simply random text. </p>
  </body>
</html>
```

Set the Font Weight

The following example demonstrates how to set the font weight of an element. The font-weight property provides the functionality to specify how bold a font is. Possible values could be *normal, bold, bolder, lighter, 100, 200, 300, 400, 500, 600, 700, 800, 900*.

```
<!DOCTYPE html>
<html>
  <head>
  </head>

  <body>
    <p style = "font-weight:bold;"> Contrary to popular belief, Lorem Ipsum is not simply
random text. </p>

    <p style = "font-weight:bolder;"> Contrary to popular belief, Lorem Ipsum is not simply
random text. </p>

    <p style = "font-weight:800;"> This font is 800 weight. </p>
  </body>
</html>
```

Set the Font Size

The following example demonstrates how to set the font size of an element. The font-size property is used to control the size of fonts. Possible values could be *xx-small, x-small, small, medium, large, x-large, xx-large, smaller, larger, size in pixels or in %*.

```
<!DOCTYPE html>
<html>
  <head>
  </head>
  <body>
    <p style = "font-size:30px;"> This font size is 30 pixels</p>
     <p style = "font-size:small;"> This font size is small  </p>
     <p style = "font-size:large;"> This font size is large </p>
  </body>
</html>
```

Set the Font Size Adjust

The following example demonstrates how to set the font size adjust of an element. This property enables you to adjust the x-height to make fonts more legible. Possible value could be any number.

```
<!DOCTYPE html>
<html>
  <head>
  </head>
  <body>
    <p style = "font-size-adjust:0.10;">  Text is using a font-size-adjust value. </p>
  </body>
</html>
```

Set the Font Stretch

The following example demonstrates how to set the font stretch of an element. This property relies on the user's computer to have an expanded or condensed version of the font being used.

Possible values could be *normal, wider, narrower, ultra-condensed, extra-condensed, condensed, semi-condensed, semi-expanded, expanded, extra-expanded, ultra-expanded*.

```
<!DOCTYPE html>
<html>
  <head>
  </head>
  <body>
    <p style = "font-stretch:ultra-expanded;"> Contrary to popular belief, Lorem Ipsum is
not simply random text. </p>
  </body>
</html>
```

Shorthand Property

You can use the *font* property to set all the font properties at once. For example −

```
<!DOCTYPE html>
<html>
  <head>
  </head>
  <body>
    <p style = "font:italic small-caps bold 15px georgia;"> Applying all the properties on the
text at once. </p>
  </body>
</html>
```

Text Effects

CSS has a lot of properties for formatting text. You can set following text properties of an element −

- The **color** property is used to set the color of a text.
- The **direction** property is used to set the text direction.
- The **letter-spacing** property is used to add or subtract space between the letters that make up a word.
- The **word-spacing** property is used to add or subtract space between the words of a sentence.
- The **text-indent** property is used to indent the text of a paragraph.
- The **text-align** property is used to align the text of a document.
- The **text-decoration** property is used to underline, overline, and strikethrough text.
- The **text-transform** property is used to capitalize text or convert text to uppercase or lowercase letters.
- The **white-space** property is used to control the flow and formatting of text.
- The **text-shadow** property is used to set the text shadow around a text.

Set the Text Color

The following example demonstrates how to set the text color. Possible value could be any color name in any valid format.

```
<!DOCTYPE html>
<html>
  <head>
  </head>
  <body>
    <p style = "color:red;">   Text will be written in red. </p>
  </body>
</html>
```

Set the Text Direction

The following example demonstrates how to set the direction of a text. Possible values are *ltr or rtl*.

```
<!DOCTYPE html>
<html>
  <head>
  </head>
  <body>
```

```
    <p style = "direction:rtl;"> Text will be rendered from right to left </p>
  </body>
</html>
```

Set the Space between Characters

The following example demonstrates how to set the space between characters. Possible values are *normal or a number specifying space.*.

```
<!DOCTYPE html>
<html>
  <head>
  </head>

  <body>
    <p style = "letter-spacing:15px;"> Text is having 15px space between letters. </p>
  </body>
</html>
```

Set the Space between Words

The following example demonstrates how to set the space between words. Possible values are *normal or a number specifying space*.

```
<!DOCTYPE html>
<html>
  <head>
  </head>
  <body>
    <p style = "word-spacing:10px;"> Text is having 10px space between words. </p>
  </body>
</html>
```

Set the Text Indent

The following example demonstrates how to indent the first line of a paragraph. Possible values are in *percentage or a number specifying indent space*.

```
<!DOCTYPE html>
<html>
  <head>
  </head>
  <body>
```

```
     <p style = "text-indent:15px;">  Text will have first line indented by 15px and this line
will remain at its actual position this is done by CSS text-indent property. </p>
   </body>
</html>
```

Set the Text Alignment

The following example demonstrates how to align a text. Possible values are *left, right, center, justify*.

```
<!DOCTYPE html>
<html>
  <head>
  </head>
  <body>
    <p style = "text-align:right;"> Text will be right aligned. </p>
    <p style = "text-align:center;"> Text will be center aligned. </p>
    <p style = "text-align:left;"> Text will be left aligned. </p>
  </body>
</html>
```

Decorating the Text

The following example demonstrates how to decorate a text. Possible values are *none, underline, overline, line-through, blink*.

```
<!DOCTYPE html>
<html>
  <head>
  </head>
  <body>
    <p style = "text-decoration:underline;"> Text will be underlined</p>
    <p style = "text-decoration:line-through;"> Text will be striked through. </p>
    <p style = "text-decoration:overline;"> Text will have a over line. </p>
    <p style = "text-decoration:blink;"> Text will have blinking effect  </p>
  </body>
</html>
```

Set the Text Cases

The following example demonstrates how to set the cases for a text. Possible values are *none, capitalize, uppercase, lowercase*.

```
<!DOCTYPE html>
<html>
  <head>
  </head>
  <body>
    <p style = "text-transform:capitalize;"> Text will be capitalized </p>
    <p style = "text-transform:uppercase;"> Text will be in uppercase </p>
    <p style = "text-transform:lowercase;">  Text will be in lowercase </p>
  </body>
</html>
```

Set the White Space between Text

The following example demonstrates how white space inside an element is handled. Possible values are *normal, pre, nowrap*.

```
<!DOCTYPE html>
<html>
  <head>
  </head>

  <body>
    <p style = "white-space:pre;">Text has a line break and the white-space pre setting
      tells the browser to honor it just like the HTML pre tag.
    </p>
  </body>
</html>
```

Set the Text Shadow

The following example demonstrates how to set the shadow around a text. Text shadow may not be supported by all the browsers or older versions of browser.

```
<!DOCTYPE html>
<html>
  <head>
  </head>
  <body>
    <p style = "text-shadow:10px 15px 10px black;"> Text will have a black shadow.</p>
  </body>
</html>
```

2.6. Colors, Background Images, and Masks

Colors

CSS uses color values to specify a color. Typically, these are used to set a color either for the foreground of an element (i.e., its text) or else for the background of the element. They can also be used to affect the color of borders and other decorative effects.

You can specify your color values in various formats. Following table lists all the possible formats −

Format	Syntax	Example
Hex Code	#RRGGBB	p{color:#FF0000;}
Short Hex Code	#RGB	p{color:#6A7;}
RGB %	rgb(rrr%,ggg%,bbb%)	p{color:rgb(50%,50%,50%);}
RGB Absolute	rgb(rrr,ggg,bbb)	p{color:rgb(0,0,255);}
keyword	aqua, black, etc.	p{color:teal;}

Hex Codes

A hexadecimal is a 6 digit representation of a color. The first two digits(RR) represent a red value, the next two are a green value(GG), and the last are the blue value(BB).

A hexadecimal value can be taken from any graphics software like Adobe Photoshop, Jasc Paintshop Pro, or even using Advanced Paint Brush.

Each hexadecimal code will be preceded by a pound or hash sign '#'.

For example: Black - #000000 , White- #FFFFFF, Blue - #0000FF

Short Hex Codes

This is a shorter form of the six-digit notation. In this format, each digit is replicated to arrive at an equivalent six-digit value. For example: #6A7 becomes #66AA77.

A hexadecimal value can be taken from any graphics software like Adobe Photoshop, Jasc Paintshop Pro, or even using Advanced Paint Brush.

Each hexadecimal code will be preceded by a pound or hash sign '#'.

For example: Black - #000, White- #FFF, Blue - #0FF

RGB Values

This color value is specified using the rgb( ) property. This property takes three values, one each for red, green, and blue. The value can be an integer between 0 and 255 or a percentage.

NOTE − All the browsers does not support rgb() property of color so it is recommended not to use it.

For example: Black - rgb(0,0,0), White- rgb(255,255,255), Blue - rgb(0,0,255)

Background Images

The **background-image** property is used to set the background image to an element. We can set the background image by calling local stored images.

```
<!DOCTYPE html>
<html>
  <head>
    <style>
      body  {
        background-image: url("/css/images/image1.jpg");
        background-color: #cccccc;
       background-repeat: repeat;

      }
    </style>
  </head>

  <body>
    <h1>Hello World!</h1>
  </body>
<html>
```

In above program we use *background-repeat* property with repeat value. This property also contains another value *no-repeat* if you don't want to repeat an image, in this case image will display only once.

By default *background-repeat* property will have *repeat* value.

Masks

The **mask** property in CSS is used to hide an element using the clipping or masking the image at specific points. Masking defines how to use an image or the graphical element as a luminance or alpha mask. It is a graphical operation that can fully or partially hide the portions of an element or object.

Using masking, it is possible to show or hide the parts of an image with different opacity levels. In CSS, the masking is achieved by using the **mask-image** property, and we have to provide an image as the mask.

Example

```
<!DOCTYPE html>
<html>
<head>
<style>
div img{
            width: 200px;
            height: 200px;
}
#masked{
             width: 300px;
             height: 300px;
            -webkit-mask-box-image: url(box.png) 20;

}
</style>
</head>
  <body>
<center>
<div id = "one">
<h2> Original Images </h2>
<img src = "forest.jpg">
<img src = "box.png">
</div>
<h2> After masking </h2>
<img src = "forest.jpg" id = "masked">
</center>
</body>
</html>
```

Example 2

```
<!DOCTYPE html>
<html>
<head>
<style>
#circle{
           width: 300px;
           height: 300px;
-webkit-mask-image: radial-gradient(circle at 50% 50%, blue 40%, rgba(0,0,0,0.3) 70%);
           border: 9px ridge red;
}
#ellipse{
            width: 300px;
            height: 300px;
 -webkit-mask-image: radial-gradient(ellipse 80% 30% at 50% 50%, blue 40%,
rgba(0,0,0,0.3) 55%);
```

```
        border: 9px ridge red;
}
</style>
</head>
 <body>
<center>
<img src = "forest.jpg" id = "circle">
<img src = "forest.jpg" id = "ellipse">
</center>
</body>
</html>
```