CASCADING STYLE SHEETS (CSS) - PART 2

RMUA Rathnayake

University of Colombo School of Computing

CSS PREPROCESSOR

CSS Preprocessor is a scripting language that extends CSS and gets compiled into regular CSS syntax, so that it can be read by your web browser.

It provides functionalities like

- *variables*,
- *functions*,
- *Mixins*
- *operations*

These functionalities allow you to build dynamic CSS.

CSS PREPROCESSOR

LESS was designed by **Alexis Sellier** in 2009. LESS is an open-source.

The first version of LESS was written in Ruby; in the later versions, the use of Ruby was replaced by JavaScript.

Features

- Cleaner and more readable code can be written in an organized way.
- We can define styles and it can be reused throughout the code.
- LESS is based on JavaScript and is a super set of CSS.
- LESS is an agile tool that sorts out the problem of code redundancy.

CSS PRE-PROCESSOR

Pre-Processor:
A computer program that modifies data to conform with the input requirements of another program

Problems with traditional CSS,

- Difficult to Maintain
- Lack of Reusability
- Lack of Extensibility

Maintainable, Reusable and Extensible set of styling instructions

WHY CSS PREPOCESSOR

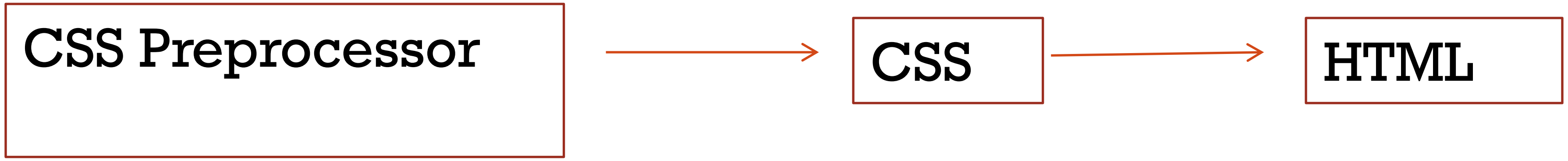
- Better code organization and readability –Can reuse stylesheet definition instructions.
- More flexible –Can add conditional statements
- Shareable –Can reuse others codes and vise versa
- Cross–browser compatible code generation

Resources:

- SAAS –https://www.tutorialspoint.com/sass/
- LESS –http://www.tutorialspoint.com/less/

CSS PRE-PROCESSOR …

A Scripting Language that extends CSS and gets compiled into regular CSS syntax.



CSS Preprocessor

CSS

HTML

CSS PRE-PROCESSOR …

Popular Preprocessors:

- LESS –NodeJSCompiler (Leaner Style Sheets(**Less**) is a backwards-compatible language extension for **CSS) – (Backward compatibility** is a property of a system, product, or technology that allows for interoperability with an older legacy system - sometimes also called downward compatibility)
- SASS –Ruby Compiler (Syntactically Awesome Style Sheets (**Sass)** is completely compatible with all versions of **CSS**. We take this compatibility seriously, so that you can seamlessly use any available **CSS** libraries.)
- Stylus –NodeJSCompiler (**Stylus** is a dynamic stylesheet language that is compiled into Cascading Style Sheets (**CSS**). Its design is influenced by Sass and LESS.)

LESS IS A CSS

LESS is a CSS pre-processor that enables customizable, manageable and reusable style sheet for website.

LESS is a dynamic style sheet language that extends the capability of CSS. LESS is also cross browser friendly.

CSS Preprocessor is a scripting language that extends CSS and gets compiled into regular CSS syntax, so that it can be read by your web browser.

It provides functionalities like *variables*, *functions*, *mixins* and *operations* that allow you to build dynamic CSS.

SASS

SASS:

SASS (Syntactically Awesome Stylesheet) is a CSS pre-processor, which helps to reduce repetition with CSS and saves time. It is more stable and powerful CSS extension language that describes the style of document structurally.

It was initially designed by **Hampton Catlin** and developed by **Natalie Weizenbaum** in 2006. Later, **Weizenbaum** and **Chris Eppstein** used its initial version to extend the Sass with SassScript.

WHY TO USE SASS?

- It is a pre-processing language which provides indented syntax (its own syntax) for CSS.

- It 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, coded in **Ruby**.

- It provides the document style in a good, structured format than flat CSS. It uses re-usable methods, logic statements and some of the built-in functions such as color manipulation, mathematics and parameter lists.

FEATURES OF SASS

- It is more stable, powerful, and compatible with versions of CSS.
- It is a super set of CSS and is based on JavaScript.
- It is known as syntactic sugar for CSS, which means it makes easier way for user to read or express the things more clearly.
- It uses its own syntax and compiles to readable CSS.
- You can easily write CSS in less code within less time.
- It is an open source pre-processor, which is interpreted into CSS.

ADVANTAGES OF SASS

- It allows writing clean CSS in a programming construct.
- It helps in writing CSS quickly.
- It is a superset of CSS, which helps designers and developers work more efficiently and quickly.
- As Sass is compatible with all versions of CSS, we can use any available CSS libraries.
- It is possible to use nested syntax and useful functions such as color manipulation, mathematics and other values.

DISADVANTAGES OF SASS

- It takes time for a developer to learn new features present in this pre-processor.
- If many people are working on the same site, then should use the same preprocessor. Some people use Sass and some people use CSS to edit the files directly. Therefore, it becomes difficult to work on the site.
- There are chances of losing benefits of browser's built-in element inspector.

SYSTEM REQUIREMENTS FOR SASS

- **Operating System** − Cross-platform
- **Browser Support** − IE (Internet Explorer 8+), Firefox, Google Chrome, Safari, Opera
- **Programming Language** − Ruby

ADVANTAGES AND DISADVANTAGES (LESS)

Advantages

- LESS easily generates CSS that works across the browsers.
- LESS enables you to write better and well-organized code by using *nesting*.
- Maintenance can be achieved faster by the use of *variables*.
- LESS enables you to reuse the whole classes easily by referencing them in your rule sets.
- LESS provides the use of *operations* that makes coding faster and saves time.

Disadvantages

- It takes time to learn if you are new to CSS preprocessing.
- Due to the tight coupling between the modules, more efforts should be taken to reuse and/or test dependent modules.
- LESS has less framework compared to older preprocessor like SASS, which consists of frameworks *Compass*, *Gravity* and *Susy*.

CSS PRE-PROCESSOR …

```
.large-heading {
font-family:Helvetica, Arial, sans-serif;
font-weight:bold;
font-size:24px;
text-transform:uppercase;
line-height:1.2em;
color:#ccc;
}
.med-heading {
font-family:Helvetica, Arial, sans-serif;
font-weight:bold;
font-size:18px;
text-transform:uppercase;
line-height:1.2em;
color:#ccc;
}
.small-heading {
font-family:Helvetica, Arial, sans-serif;
font-weight:bold;
font-size:14px;
text-transform:uppercase;
line-height:1.2em;
color:#ccc;
}
```

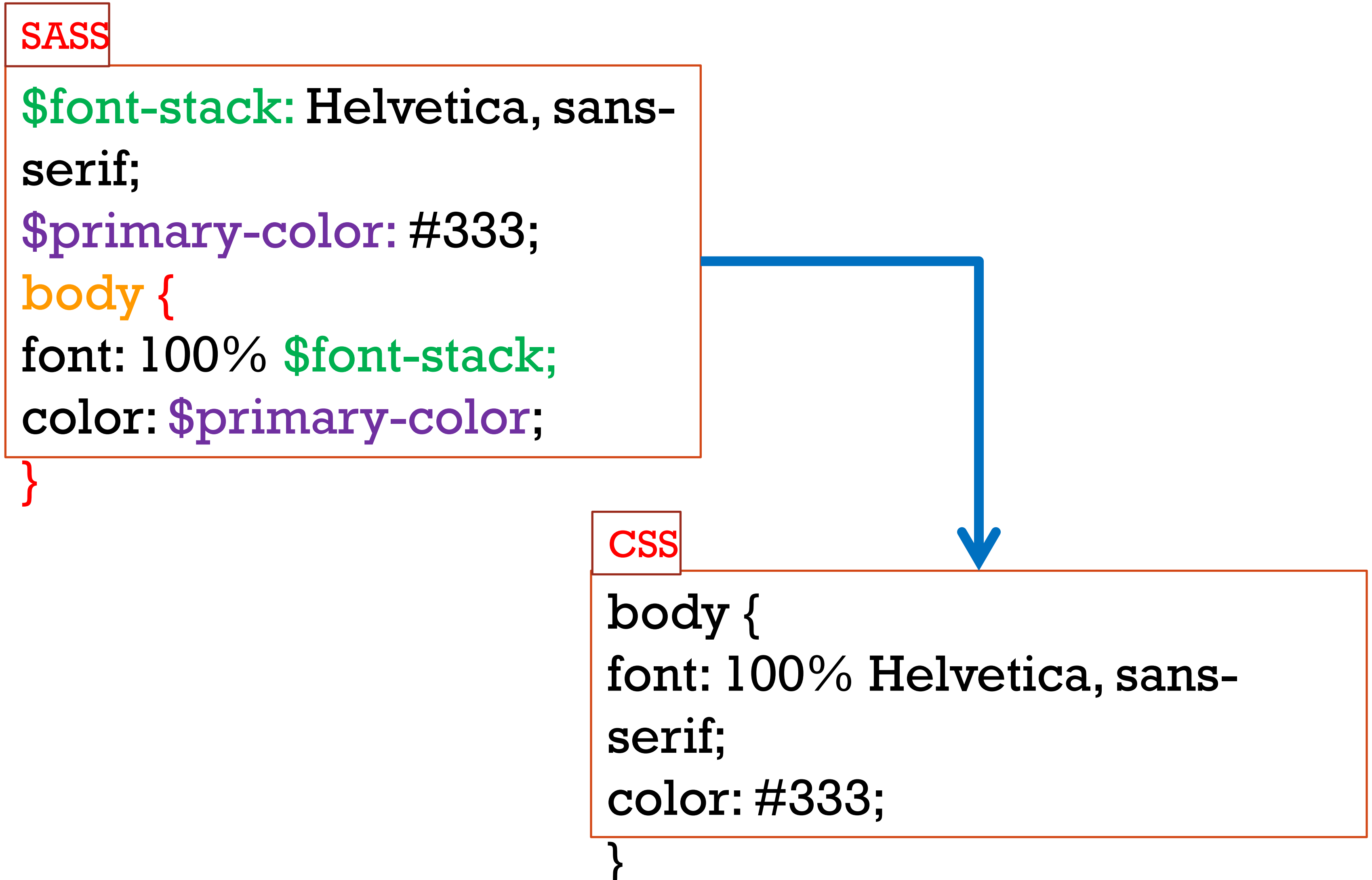
CSS

```
.large-heading {
font-family:Helvetica, Arial, sans-serif;
font-weight:bold;
font-size:24px;
text-transform:uppercase;
line-height:1.2em;
color:#ccc;
}
.med-heading {
.large-heading;
font-size:18px;
}
.small-heading {
.large-heading;
font-size:14px;
}
```

LESS

VARIABLES

We can use variables to store repeatedly used styling parameters and do some manipulations with it.



SASS

```
$font-stack: Helvetica, sans-serif;
$primary-color: #333;
body {
font: 100% $font-stack;
color: $primary-color;
}
```

CSS

```
body {
font: 100% Helvetica, sans-serif;
color: #333;
}
```

NESTING

SASS

```
nav{
ul{
margin: 0;
padding: 0;
list-style: none;
}
li { display: inline-block;
}
a {
display: block;
padding: 6px 12px;
text-decoration: none;
}
}
```

CSS

```
navul{
margin: 0;
padding: 0;
list-style: none;
}
navli {
display: inline-block;
}
nava {
display: block;
padding: 6px 12px;
text-decoration: none;
}
```

MIXINS

Grouping of multiple code lines together that can then be reused throughout the stylesheet.

(**Mixins** allow document authors to define patterns of property value pairs, which can then be reused in other rule sets)

SASS

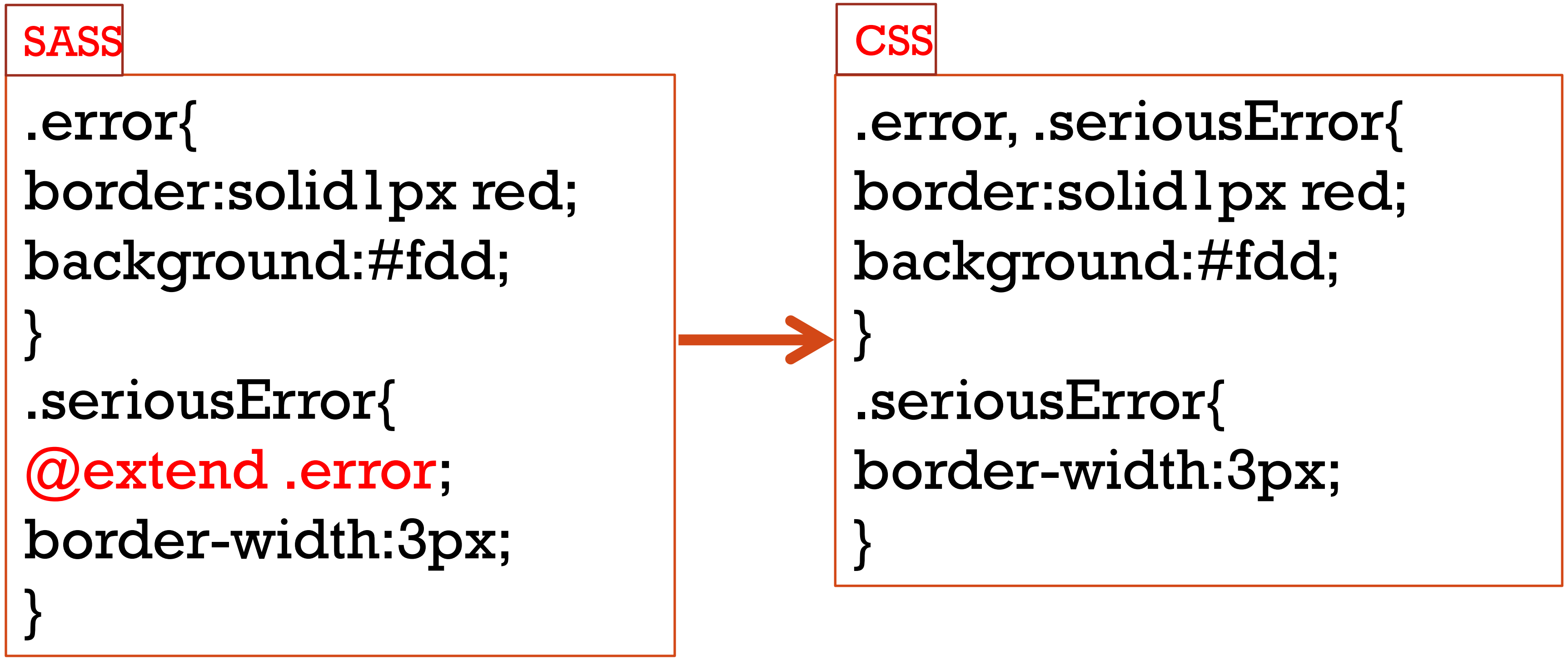
```
@mixin notification {
padding:10px;
border-radius:5px;
font-size:1em;
}
.error{
@include notification;
background:red;
color:white;
}
```

CSS

```
.error{
padding:10px;
border-radius:5px;
font-size:1em;
background:red;
color:white;
}
```

EXTEND

- Using @extend enables a selector to inherit the rules of another selector.



SASS

```
.error{
border:solid1px red;
background:#fdd;
}
.seriousError{
@extend .error;
border-width:3px;
}
```

CSS

```
.error, .seriousError{
border:solid1px red;
background:#fdd;
}
.seriousError{
border-width:3px;
}
```

CONTROL INSTRUCTIONS

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 mixing and functions to write small algorithms to make writing your Sass easier.

Sass supports 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.

CONTROL INSTRUCTIONS

SASS

```
@if lightness($color) >
30% {
background-color:
black;
}
@else {
background-color:
white;
}
```

If-Else

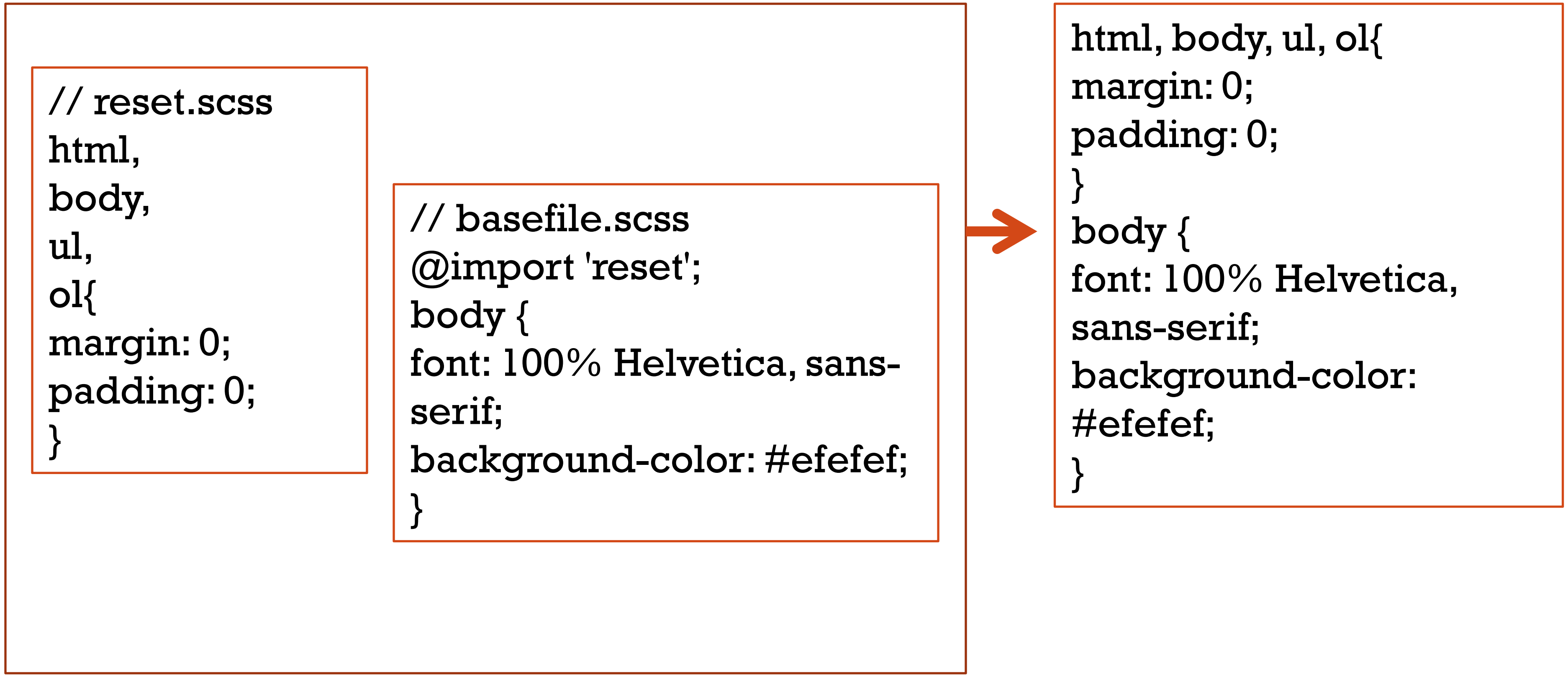
CSS

```
@for $ifrom 1px to 3px {
.border-#{i} {
border: $isolid blue;
}
}
```

For Loop

https://sass-lang.com/documentation

IMPORT



```
// reset.scss
html,
body,
ul,
ol{
margin: 0;
padding: 0;
}
```

```
// basefile.scss
@import 'reset';
body {
font: 100% Helvetica, sans-serif;
background-color: #efefef;
}
```

```
html, body, ul, ol{
margin: 0;
padding: 0;
}
body {
font: 100% Helvetica, sans-serif;
background-color: #efefef;
}
```

BORDER-STYLE

```
<style type="text/css">
   p.dotted {border-style: dotted}
   p.dashed {border-style: dashed}
   p.solid {border-style: solid}
   p.double {border-style: double}
   p.groove {border-style: groove}
   p.ridge {border-style: ridge}
   p.inset {border-style: inset}
   p.outset {border-style: outset}
</style>
```

```
<body>
   <p class="dotted">A dotted border</p>
   <p class="dashed">A dashed border</p>
   <p class="solid">A solid border</p>
   <p class="double">A double border</p>
   <p class="groove">A groove border</p>
   <p class="ridge">A ridge border</p>
   <p class="inset">An inset border</p>
   <p class="outset">An outset border</p>
</body>
```

https://www.w3schools.com/cssref/

DIV ELEMENT WITH FLOAT

```
<style type="text/css">
div
{
float:right;
width:120px;
margin:0 0 15px 20px;
padding:15px;
border:1px solid black;
text-align:center;
}
</style>
<body>
<div>
<img src="logocss.gif" width="95" height="84" /><br />
CSS is fun!
</div>
<p>
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
</p>
<p>
In the paragraph above, the div element is 120 pixels wide and it contains the image.
The div element will float to the right.
Margins are added to the div to push the text away from the div.
Borders and padding are added to the div to frame in the picture and the caption.
</p>
```

FLOAT WITH MENU

```
<head>
<style type="text/css">
ul
{
float:left;
width:100%;
padding:0;
margin:0;
list-style-type:none;
}
a
{
float:left;
width:6em;
text-decoration:none;
color:white;
background-color:purple;
padding:0.2em 0.6em;
border-right:1px solid white;
}
a:hover {background-color:#ff3300}
li {display:inline}
</style>
</head>
<body>
<ul>
<li><a href="#">Link one</a></li>
<li><a href="#">Link two</a></li>
<li><a href="#">Link three</a></li>
<li><a href="#">Link four</a></li>
</ul>
```

DEFINE SEMANTIC ELEMENTS AS BLOCK ELEMENTS

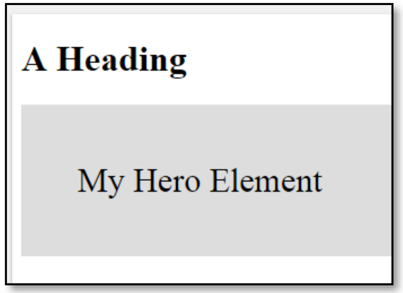
- HTML5 defines eight new **semantic** elements.
- All these are **block-level** elements.
- To secure correct behavior in older browsers, you can set the CSS **display** property for these HTML elements to **block**:

```
header, section, footer, aside, nav, main, article,
figure {
   display: block;
}
```

ADD NEW ELEMENTS TO HTML

This example adds a new element called **<myHero>** to an HTML page, and defines a style for it:

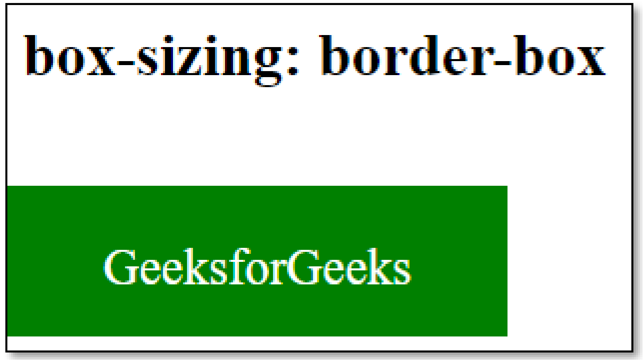
```
<!DOCTYPE html>
<html>
<head>
 <script>document.createElement("myHero")</script>
 <style type=“text/css”>
 myHero {
    display: block;
    background-color: #dddddd;
    padding: 50px;
    font-size: 30px;
 }
 </style>
</head>
<body>

<h1>A Heading</h1>
<myHero>My Hero Element</myHero>
</body>
</html>
```



A Heading

My Hero Element

BOX-SIZING: BORDER-BOX



box-sizing: border-box

GeeksforGeeks

https://www.geeksforgeeks.org/css-box-sizing-property/

```
<html>
  <head>
    <title>box-sizing Property</title>
    <style>
      div {
        width: 200px;
        height: 60px;
        padding: 20px;
        border: 2px solid green;
        background: green;
        color: white;
      }
      .border-box {
        box-sizing: content-box;
      }
    </style>
  </head>
  <body style = "text-align: center;">
    <h2>box-sizing: border-box</h2>
    <br>
    <div class="border-
box">GeeksforGeeks</div>
  </body> </html>
```

WILDCARD SELECTOR

Wildcard Selectors (*, ^ and $) in CSS for classes

Wildcard selector is used to select multiple elements simultaneously. It selects similar type of class name or attribute and use CSS property. * wildcard also known as containing wildcard.

[attribute*=”str”] Selector: The [attribute*=”str”] selector is used to select that elements whose attribute value contains the specified sub string *str*.

This example shows how to use a wildcard to select all div with a class that contains *str*. This could be at the start, the end or in the middle of the class.

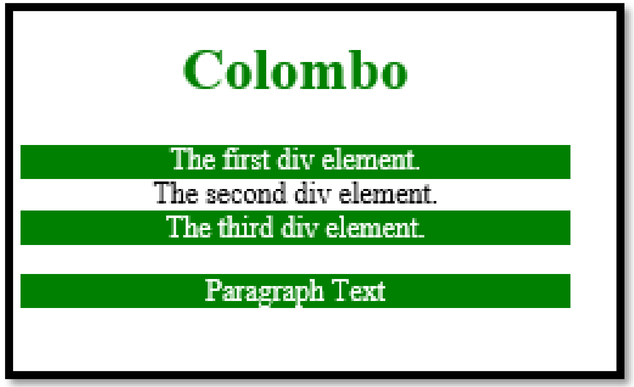
Syntax:

```
[aattribute*="value"] { // CSS property }
```

Example:

EXAMPLE

```
<!DOCTYPE html>
<html>
  <head>
    <style>

      /* Define styles of selected items, h1 and
        rest of the body */
      [class*="str"] {  /* WE USE * HERE */
        background: green;
        color: white;
      }
      h1 {
        color:green;
      }
      body {
        text-align:center;
        width:60%;
      }
    </style>
  </head>
  <body>
    <h1>Colombo</h1>

    <!-- Since we have used * with str, all items with
      str in them are selected -->
    <div class="first_str">The first div element.</div>
    <div class="second">The second div element.</div>
    <div class="my-strt">The third div element.</div>
    <p class="mystr">Paragraph Text</p>
  </body>
</html>
```



Colombo

The first div element.

The second div element.

The third div element.

Paragraph Text

https://www.geeksforgeeks.org/wildcard-selectors-and-in-css-for-classes/

[ATTRIBUTE^=”STR”] SELECTOR:

Syntax:

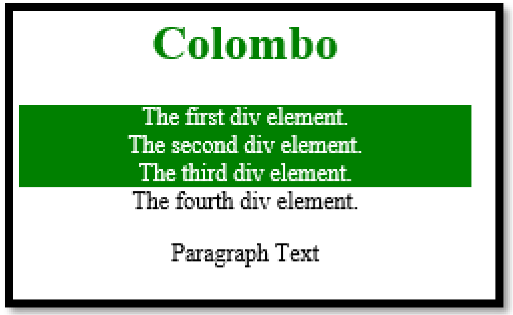
```
[attribute^="str"] { // CSS property }
```

Example:

```
<!DOCTYPE html>
<html>
    <head>
        <style>
            [class^="str"] { /*WE USE ^ HERE */
                background: green;
                color: white;
            }
            h1 {
                color:green;
            }
            body {
                text-align:center;
                width:60%;
            }
        </style>
    </head>
    <body>
        <h1>Colombo</h1>

        <!-- All items beginning with str are highlighted -->
        <div class="strfirst">The first div element.</div>
        <div class="strsecond">The second div element.</div>
        <div class="str-start">The third div element.</div>
        <div class="end-str">The fourth div element.</div>
        <p class="my">Paragraph Text</p>
    </body>
</html>
```

[attribute^=”str”] Selector: The [attribute^=”value”] selector is used to select those elements whose attribute value begins with a specified value *str*.

This example shows how to use a wildcard to select all div with a class that starts with *str*.



Colombo

The first div element.
The second div element.
The third div element.
The fourth div element.

Paragraph Text

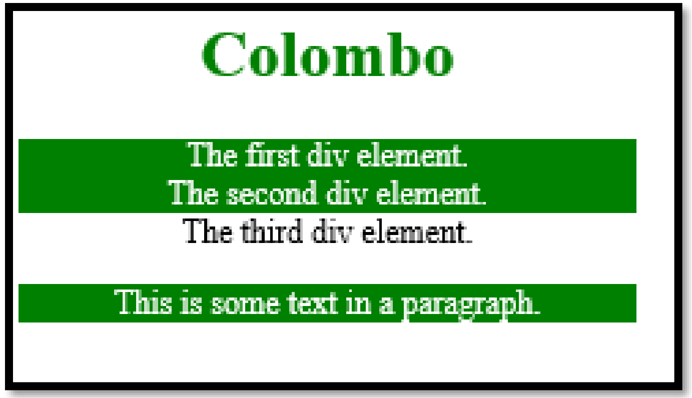
ATTRIBUTE$=”STR”] SELECTOR:

[attribute$=”str”] Selector: The [attribute$=”value”] selector is used to select those elements whose attribute value ends with a specified value *str*. The following example selects all elements with a class attribute value that ends with *str*.

Syntax:

[attribute$="str"] { // CSS property }

```
<!DOCTYPE html>
<html>
  <head>
    <style>
      [class$="str"] { /* WE USE $ HERE */
        background: green;
        color: white;
      }
      h1 {
        color:green;
      }
      body {
        text-align:center;
        width:60%;
      }
    </style>
  </head>
  <body>
    <h1>Colombo</h1>

    <!-- All items ending with str are highlighted -->
    <div class="firststr">The first div element.</div>
    <div class="stsecondstr">The second div
element.</div>
    <div class="start">The third div element.</div>
    <p class="mystr">This is some text in a
paragraph.</p>
  </body>
</html>
```



Colombo

The first div element.
The second div element.
The third div element.

This is some text in a paragraph.

CLEAR: BOTH;

both: No floating elements allowed on either side.
clear: both;

When you use clear: both;, you're instructing the browser that the element should move below any preceding elements that are floated to the left or the right.

This is particularly useful when you want to ensure that an element appears below any floated elements, effectively clearing the float.

```
#footer
{
		clear: both;
		margin: 0;
		padding: .5em;
		color: #333;
		background-color: #F0FFFF;
		border-top: 1px solid gray;
}
```