Cascading Style Sheets (CSS)

Dr. Samantha Mathara Arachchi
B.Sc,Pg.Dip(Com.Tech.),Pg.Dip.(IM),M.Sc.(IM),PhD(My.),SEDA(UK),MCS(SL) MACM,MIEEE IEEE

e-mail:(ssp@ucsc.cmb.ac.lk)

University of Colombo School of Computing

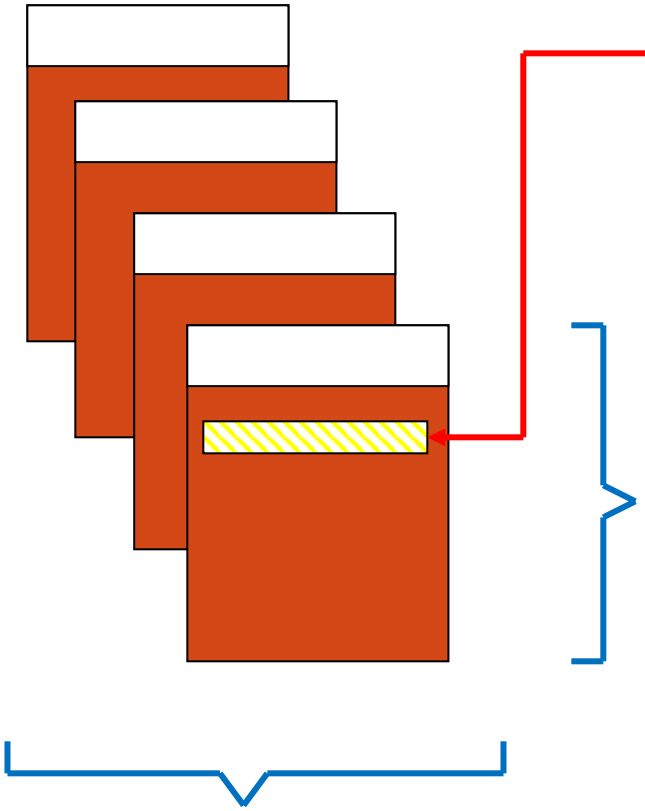
Cascading Style Sheet (CSS)

- Recommended by W3C
- The Characteristics of CSS
  - The layout of the page can be flexibly
    - It can specify the font name and size precisely
    - It can specify margin and indent
    - It can specify the position of the text and image
  - The page and web layout can be managed collectively
    - The changes can be done easily
- To validate your style sheet
  - http://jigsaw.w3.org/css-validator/validator-uri.html

Implementation of CSS

- Methods of Implementation
  - Following are the 4 methods of implementing the css.
    - **Inline Style sheet** (Specify style directly by using the style attributes)
    - **Embedded style sheet** (Define style in advance to STYLE element, then apply
    - **Linking style sheet** (By using LINK elements link the external file where style has been defined
    - **Import style sheet** (By using STYLE element, specify the external file (define style) to be imported

Case by case example



When you want to specify style only at this position

Specify style directly with **[Inline style sheet]**

When you want to specify the common style only on this page

Define and apply style with **[Embedded style sheet]**

When you want to specify the common style on all the pages

Define and apply style in external file **[Linking/Import]**

Inline Style Sheet

- **Specify style directly by using STYLE attributes toward each element.**

```
<BODY>
<Tag STYLE="property:value"> - </Tag>
</BODY>
```

```
<BODY>
<H1 STYLE="color: red">Red heading 1 </H1>
<P STYLE="color: blue; FONT-size:20px"> Blue Paragraph</P>
    ⋮
</BODY>
```

Separator

- Use for each element **within the BODY**
- At STYLE attribute, specify the style to use
- Multiple styles can be defined, separated with semi-colon.
- The are where the style is applied is different depending on the element

Embedded Style Sheet

- Define the style within the HEAD, then apply the style in the BODY, style is defined with the form of [Rule]

```
<HEAD>
<STYLE TYPE="TEXT/CSS">
</STYLE>
<HEAD>
```

Selector {Property:value}

Selector: Tie up the HTML element and style defined by definition part

Property: Specify the property toward the specified element in selector

Value: Specify the applied value to the style

```
<HEAD>
<STYLE TYPE="TEXT/CSS">
H1{color:red; font-size:20px}
</STYLE>
</HEAD>
<BODY>
<H1>Heading</H1>
⋮
</BODY>
```

Definition

Applied

Linking Style Sheet

- Link the style and the external file which defines the style within the HEAD.
- File Extension is **.CSS**

```
<HEAD>
<LINK REL="stylesheet" TYPE="text/css" HREF="url">
</HEAD>
```

style.css - Notepad

File Edit Format View Help

```
H1{fonr-family:"Times New Roman";
  font-size:36px}
```

```
<HEAD>
<TITLE>title</TITLE>
<LINK REL="stylesheet" TYPE="text/css" HREF="style.css">
</HEAD>
<BODY>
<H1>Heading</H1>

</BODY>
```

Style file defining style

Linking Style Sheet …

- In REL attribute, specify the relationship with the file linked.
- In TYPE attribute, specify the MIME type of style file
- In HREF attribute, specify the style **file location and name.** (file extension is .css)
- Both absolute path and relative path can be specified for the style file name
- Define only the [rules] in style file
- Applied **in BODY part**

Import Style Sheet

- Import the external file where the style has been defined **in HEAD part**.

```
<HEAD>
<STYLE TYPE="TEXT/CSS">
                @import url (file name or URL);
</STYLE>
</HEAD>
```

The Priority among 4 Implementation Methods

- When multiple styles are specified in the document, the following priority order shall be applied.

Style sheet by external files (**Linking/Import**) < **Embedded** Style sheet < **Inline** Style sheet

- Define the general style of the Web by **[Style sheet by external files]**
- Define the style of whole page by **[Embedded style sheet]**
- Define individual style by **[Inline style sheet]**

The Selector

- If a element is used in selector, then **all style are applied in that element.**
- Styles can be specified in details using the following **5 methods.**
  - **Element Selector**
    - Always specify common style toward the element
  - **Class Selector**
    - Create and define optional name to the specify style, then apply it.
  - **ID Selector**
    - Create and define the optional name towards the specify style, and apply it at one place in a document
  - **Group Selector**
    - Apply the common style to multiple elements
  - **Context Selector**
    - Apply style only specified part where multiple elements are all specified.

1. Element Selector

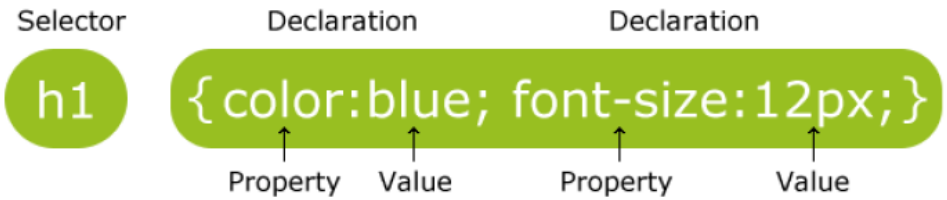
- The **common style can be applied to an element at all time**
- In Selector, **specify the element name** to apply the style

```
<HEAD>
        <STYLE TYPE="text/css">
         Element {Property:Value}
        </STYLE>
</HEAD>
```

```
<HEAD>
        <STYLE TYPE="text/css">
        H1{color:red}
        H2{color:blue}
        </STYLE>
</HEAD>
<BODY>
        <H1>Heading 1</H1>
        <H2>Item 1</H2>

        <H1>Heading 2</H1>
        <H2>Item 2</H2>
</BODY>
```

Definition / declaration



Applied

2. Class Selector

- In TYPE attribute, specify the MIME type of the style definition part.
- **Class name starts with a period (.)**
- In **Selector, specify the name created for the define style**
- When **applying the CLASS, remove the period (.)**

```
<HEAD>
        <STYLE TYPE="text/css">
         .Class name {Property:Value}
        </STYLE>
</HEAD>
```

Define

```
<HEAD>
        <STYLE TYPE="text/css">
        .red{color:#FF0000}
        .blue{color:#0000FF}
        </STYLE>
</HEAD>
<BODY>
        <H1 CLASS="red">Heading 1</H1>
        <H2 CLASS="blue">Item 1</H2>

        <H1>Heading 2</H1>
        <P CLASS="red"> Paragraph </P>
</BODY>
```

Applied

Do not Apply

Applied

3. ID Selector

- Define and name a specific style, and apply it by specifying the name.
- However it cannot be called twice but some browsers can use it like the class selector for more than twice (using function)

```
<HEAD>
        <STYLE TYPE="text/css">
         #ID {Property:Value}
        </STYLE>
</HEAD>
```

Define

```
<HEAD>
        <STYLE TYPE="text/css">
        #id123{color:red}
        </STYLE>
</HEAD>
<BODY>
        <H1 ID="id123">Heading 1</H1>
        <P>Content 1</P>

</BODY>
```

When ID is specified, the "sharp" will be removed

Applied

4. Grouped Selector

- By grouping multiple elements, **separated with comma (,)**, common style can be applied.

```
<HEAD>
        <STYLE TYPE="text/css">
         Element 1, Element 2,…. {Property:Value}
        </STYLE>
</HEAD>
```

```
<HEAD>
        <STYLE TYPE="text/css">
        H1,H2,H3 {font:24px;
                  color:blue}
        </STYLE>
</HEAD>
<BODY>
        <H1>Heading 1</H1>
        <H2>Heading 1</H2>
        <H3>Heading 1</H3>

</BODY>
```

Define

Applied

5. Context Selector

- A specific style can be applied only when multiple elements are specified simultaneously.
- In Selector, list the **elements separated with blank**

```
<HEAD>
        <STYLE TYPE="text/css">
         Element 1 Element 2 …. {Property:Value}
        </STYLE>
</HEAD>
```

```
<HEAD>
        <STYLE TYPE="text/css">
        H1 I {color: red}
        </STYLE>
</HEAD>
<BODY>
        <H1>Heading 1
        <I> 1</I>
        </H1>

</BODY>
```

Define

Applied

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

Resource:

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

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

https://www.tutorialspoint.com/sass/sass_installation.htm

CSS Pre-Processor ...

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

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.)

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* and *operations* that allow you to build dynamic CSS.

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.

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.

Advantages and Disadvantages (LSS)

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

Benefits

- Variables
- Nested Syntax
- Partials & Imports
- Mixins
- Extend/ Inheritance
- Operators
- Functions

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

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

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

Import

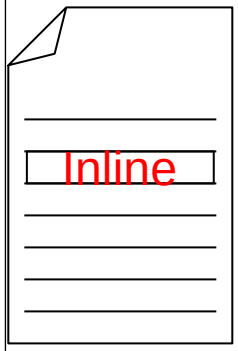
```
// _reset.scss
html,
body,
ul,
ol{
margin: 0;
padding: 0;
}
```

```
// 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;
}
```

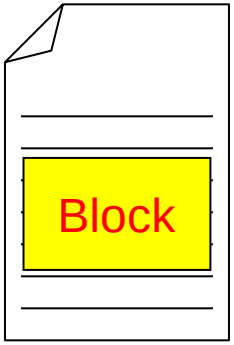
SPAN Element and DIV Element

- When the style sheet is applied only to the part of the document, it is convenient to use the following elements.
  - **SPAN** element specifies the range of inline level.
  - **DIV** element does the range of block level
- Inside DIV element can be applied SPAN element




```
<SPAN STYLE =color:red>
Inline level is red
</SPAN>
```

```
<SPAN CLASS =red>
Inline level is red
</SPAN>
```




```
<DIV STYLE =color:blue>
Block level is blue
</DIV>
```

```
<DIV CLASS =blue>
Block level is blue
</DIV>
```

<H1 style="color:red"> Hello I am Samantha </H1>

<H1 style="color:red"> Hello I am <SPAN style="color:blue">Samantha</SPAN></H1>

<H1 style="color:red"> Hello I am <DIV style= "color:blue">Samantha</DIV></H1>

Use of the float property

```
<html>
<head>
<style type="text/css">
img {
float:right
}
</style>
</head>
<body>
<p>In the paragraph below, we have added an image with style <b>float:right</b>.
   The result is that the image will float to the right in the paragraph.</p>
<p><img src="logocss.gif" width="95" height="84" />
```

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>
```

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>
```

CSS Combinators

A combinator is something that explains the relationship between the selectors.

A CSS selector can contain more than one simple selector. Between the simple selectors, we can include a combinator.
There are four different combinators in CSS3:

1. descendant selector (space)
2. child selector (>)
3. adjacent sibling selector (+)
4. general sibling selector (~)

Descendant Selector (Space)

The descendant selector matches all elements that are descendants of a specified element.
The following example selects all <p> elements inside elements:

Example

```
div p {
    background-color: yellow;
}
```

Descendant Selector …

```
<!DOCTYPE html>
<html>
<head>
<style>
div p {
    background-color: yellow;
}
</style>
</head>
<body>

<div>
  <p>Paragraph 1 in the div.</p>
  <p>Paragraph 2 in the div.</p>
  <span><p>Paragraph 3 in the div.</p></span>
</div>

<p>Paragraph 4. Not in a div.</p>
<p>Paragraph 5. Not in a div.</p>

</body>
</html>
```

Paragraph 1 in the div.

Paragraph 2 in the div.

Paragraph 3 in the div.

Paragraph 4. Not in a div.

Paragraph 5. Not in a div.

<p> elements inside elements

Child Selector (>)

The child selector selects all elements that are the immediate children of a specified element.
The following example selects all <p> elements that are immediate children of a element:
Example

```
div > p {
    background-color: yellow;
}
```

Child Selector ...

```
<!DOCTYPE html>
<html>
<head>
<style>
div > p {
    background-color: yellow;
}
</style>
</head>
<body>

<div>
  <p>Paragraph 1 in the div.</p>
  <p>Paragraph 2 in the div.</p>
  <span><p>Paragraph 3 in the div.</p></span>
<!-- not Child but Descendant -->
</div>

<p>Paragraph 4. Not in a div.</p>
<p>Paragraph 5. Not in a div.</p>

</body>
</html>
```

Paragraph 1 in the div.

Paragraph 2 in the div.

Paragraph 3 in the div.

Paragraph 4. Not in a div.

Paragraph 5. Not in a div.

<p> elements that are immediate children of a element:

Adjacent Sibling Selector (+)

- The adjacent sibling selector selects all elements that are the adjacent siblings of a specified element.
- Sibling elements must have the same parent element, and "adjacent" means "immediately following".
- The following example selects all <p> elements that are placed immediately after elements:

Example

```
div + p {
    background-color: yellow;
}
```

Adjacent Sibling Selector …

```
<!DOCTYPE html>
<html>
<head>
<style>
div + p {
    background-color: yellow;
}
</style>
</head>
<body>

<div>
  <p>Paragraph 1 in the div.</p>
  <p>Paragraph 2 in the div.</p>
</div>

<p>Paragraph 3. Not in a div.</p>
<p>Paragraph 4. Not in a div.</p>

</body>
</html>
```

Paragraph 1 in the div.

Paragraph 2 in the div.

Paragraph 3. Not in a div.

Paragraph 4. Not in a div.

all <p> elements that are placed immediately after elements

General Sibling Selector (~ tilde)

The general sibling selector selects all elements that are siblings of a specified element.
The following example selects all <p> elements that are siblings of elements:
Example

```
div ~ p {
    background-color: yellow;
}
```

General Sibling Selector …

```
<!DOCTYPE html>
<html>
<head>
<style>
div ~ p {
    background-color: yellow;
}
</style>
</head>
<body>
<p>Paragraph 1.</p>
<div>
  <code>Some code.</code>
  <p>Paragraph 2.</p>
</div>
<p>Paragraph 3.</p>
<code>Some code.</code>
<p>Paragraph 4.</p>
</body>
</html>
```

Paragraph 1.

Some code.

Paragraph 2.

Paragraph 3.

Some code.

Paragraph 4.

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>
 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>
```

CSS Validation

- http://jigsaw.w3.org/css-validator/

References

- http://www.w3schools.com/css/css_examples.asp
- http://www.webcredible.co.uk/
- http://stylus-lang.com/
- http://oocss.org/spec/css-mixins.html
- https://sass-lang.com/
- http://lesscss.org/
- SAAS –https://www.tutorialspoint.com/sass/
- LESS –http://www.tutorialspoint.com/less/

Alternative Styles

- <link href="css/default.css" rel="stylesheet" type="text/css" title="Default" />
- <link href="css/black.css" rel="alternate stylesheet" type="text/css" title="High Contrast" />

❑**rel** = ***link-types*** **[CI]**

❑This attribute describes the relationship from the current document to the anchor specified by the href attribute. The value of this attribute is a space-separated list of link types.

❑This specification allows authors to specify a preferred style sheet as well as alternates that target specific users or media.

❑User agents should give users the opportunity to select from among alternate style sheets or to switch of off style sheets altogether.