CSS

Introduction

- **CSS** stands for **C**ascading **S**tyle **S**heets.
- The primary intention of CSS is to separate visual presentation from document content written in markup language.
- CSS describes **how HTML elements are to be displayed on screen, paper, or in other media**.
- CSS **saves a lot of work**.
- It can control the layout of multiple web pages all at once

Origin of CSS

- HTML was originally designed as a simple way of presenting information intended for sharing scientific documents and research papers online.
- Later on, as the Internet expanded from the academic and research world into the mainstream, and became more media oriented, the presentation of the web pages has become considerably important for a website's success.
- To improve web presentation capabilities CSS was introduced by W3C World wide Web Consortium It was intended to allow web designers to define the look and feel of their web pages, and to separate content from document's layout.

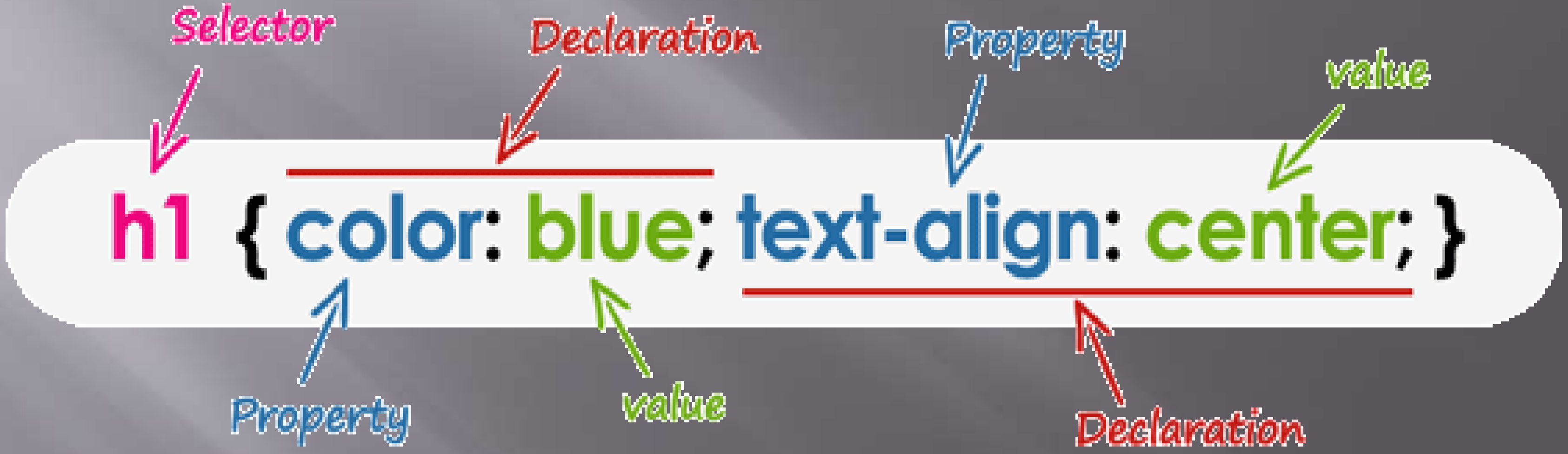
Advantages of CSS

- Allows separating content of an html document from the style and layout of that document.
- Make documents much easier to maintain and give much better control over the layout of your web pages

Some More….

- CSS Save lots of time
- Easy maintenance
- Pages load faster
- Superior styles to HTML
- Multiple Device Compatibility

CSS Syntax


Selector
Declaration
Property
value
h1 { color: blue; text-align: center; }
Property
value
Declaration

Description

- The selector points to the HTML element you want to style.
- The declaration block contains one or more declarations separated by semicolons.
- Each declaration includes a CSS property name and a value, separated by a colon.
- A CSS declaration always ends with a semicolon, and declaration blocks are surrounded by curly braces.

How to specify?

- CSS can be specified in three ways
  - Inline
    - An inline style may be used to apply a unique style for a single element.
  - Embedded/Internal
    - An internal style sheet may be used if one single page has a unique style.
  - External
    - An external style sheet, you can change the look of an entire website by changing just one file!

Inline CSS

- You can also embed your CSS code in HTML document.
- Example: `<p style=“font-family: monospace;”>`

INTERNAL CSS

- `<style></style>` always placed between `<head></head>` tags.
- Example:

```
<style>
 p { line-height: 120%; }
</style>
```

EXTERNAL CSS FILE

External CSS file will always place between <HEAD></HEAD>

```
<link rel=“stylesheet” type=“text/css”   href=“main.css” />
```

Cascade order

- What style will be used when there is more than one style specified for an HTML element?

- All the styles will "cascade" into a new "virtual" style sheet by the following rules, where number one has the highest priority:
  - Inline style (inside an HTML element)
  - External and internal style sheets (in the head section)
  - Browser default

CSS Comments

- Comments are used to explain the code, and may help when you edit the source code at a later date.
- Comments are ignored by browsers.
- A CSS comment starts with /* and ends with */. Comments can also span multiple lines

Font

- Allows you to set various styles for fonts likes font family, size and boldness, variant, etc. of a text.
- Several properties for styling fonts of the text content
  - font-family,
  - font-style,
  - font-variant,
  - font-weight and
  - font-size

Font-Family

- Two types of font family names:
  - **generic family** - a group of font families with a similar look (like "Serif" or "Monospace")
  - **font family** - a specific font family (like "Times New Roman" or "Arial")
- Allows you to set a prioritized list of font family name and/or generic family name for the text content of a selected element.
- It should hold several font names as a “fallback” system.

Syntax

```
p {
    font-family: "Times New Roman", Times, serif;
  }
```

Note: If the name of a font family is more than one word, it must be in quotation marks,   like: "Times New Roman".

Font-style

- Sets the font style for the text content of an element.
- The possible values for this property are:
  - Normal – shown normally
  - italic – shown in italic
  - Oblique – text is “leaning”

Font-size

- Sets the size of the text.
- The font-size value can be an absolute, or relative size.

- Absolute size:
  - Sets the text to a specified size
  - Does not allow a user to change the text size in all browsers (bad for accessibility reasons)
  - Absolute size is useful when the physical size of the output is known
- Relative size:
  - Sets the size relative to surrounding elements
  - Allows a user to change the text size in browsers

Relative Font-size

- Generally font size can be specified with
  - Pixels
    - use pixel units in web documents in order to produce a pixel-perfect representation of their site as it is rendered in the browser
  - Em
    - “em” is a scalable unit that is used in web document media. An em is equal to the current font-size.
  - Percentages
    - The percent unit is much like the “em” unit, save for a few fundamental differences. First and foremost, the current font-size is equal to 100% (i.e. 12pt = 100%).
  - Points
    - Points are traditionally used in print media (anything that is to be printed on paper, etc.).

Absolute Font-size

- Absolute size is specified using one of the following keywords:
  - xx-small,
  - x-small,
  - small,
  - medium,
  - large,
  - x-large,
  - xx-large.

Font Weight

- Specifies the weight or boldness of the font.
- Font-weight property are
  - normal,
  - bold,
  - bolder,
  - lighter,
  - inherit
  - 100 - 900

Font Variant

- The font-variant property allows the text to be displayed in a special small-caps variation.
- Possible ways
  - Normal
  - small-caps
    - all lowercase letters are converted to uppercase letters. However, the converted uppercase letters appears in a smaller font size than the original uppercase letters in the text.
  - Initial
    - Used to set a CSS property to its default value.
  - Inherit
    - Specifies that a property should inherit its value from its parent element.

Property	Description
font	Sets all the font properties in one declaration
font-family	Specifies the font family for text
font-size	Specifies the font size of text
font-style	Specifies the font style for text
font-variant	Specifies whether or not a text should be displayed in a small-caps font
font-weight	Specifies the weight of a font

Text

- Allows you to define several text styles such as
  - color,
  - alignment,
  - spacing,
  - decoration,
  - Transformation

- These properties give you precise control over the visual appearance of the characters, spaces, words, paragraphs, etc.

Text-color

- The color property is used to set the color of the text.
- With CSS, a color is most often specified by:
  - a color name - like "red"
  - a HEX value - like "#ff0000"
  - an RGB value - like "rgb(255,0,0)"

Text Alignment

- text-align property is used to set the horizontal alignment of a text
- Possible values for this property are:
  - left,
  - right,
  - center,
  - justify,
  - Inherit,
  - Start
    - The last line is aligned at the beginning of the line
  - End
    - The last line is aligned at the end of the line
  - Initial,
  - Inherit

Text Decoration

- Used to set or remove decorations from text.
- Possible values of this property are:
  - none
    - Defines normal text. Default text
  - underline,
    - Defines line below the text
  - overline
    - Defines the line above the text
  - line-through
    - Defines the line through the text
  - Blink
    - Allows to blink
  - Inherit

Text-decoration

- The text-decoration-style property specifies how the line, if any, will display.

Value	Description
Solid	Default value. The line will display as a single lone
double	The line will display as a double line
dotted	The line will display as a dotted line
dashed	The line will display as a dashed line
wavy	The line will display as a wavy line
initial	Sets this property to its default value.
inherit	Inherits this property from its parent element.

Text-Decoration

- The text-decoration-color property specifies the color of the text-decoration (underlines, overlines, linethroughs)
- The text-decoration-line property specifies what type of line, if any, the decoration will have.

- **Note:** You can also set the text-decoration-line using the text-decoration property, which is a short-hand property for the text-decoration-line, text-decoration-style, and the text-decoration-color properties.

Text Transformation

- Used to set the cases for a text.
- Possible values are
  - none-
  - lowercase
  - uppercase
  - capitalize

Text Indentation

- Used to specify the indentation of the first line of a text.
- Possible values for the text-indent property are:
  - percentage (%)-
    - Defines the indentation in % of the width of the parent element
  - length (specifying indent space)
    - Defines a fixed indentation in px, pt, cm, em, etc. Default value is 0
  - Inherit
  - Initial

Letter Spacing

- Used to specify the space between the characters in a text.
- increases or decreases the space between characters in a text.
- Possible values:
  - Normal
  - Length
  - Initial
  - Inherit

Line Height

- Used to specify the space between lines.
- Possible values
  - Normal
  - Number
    - A number that will be multiplied with the current font size to set the line height
  - Length
    - A fixed line height in px, pt, cm, etc.
  - Percentage
    - A line height in percent of the current font size
  - Initial
  - Inherit

Text Direction

- Used to change the text direction of an element.
  - Rtl – right to left
  - Ltl – left to right (default)

Word Spacing

- Used to specify the space between the words in a text.
- Possible values:
  - Normal
    - Defines normal space between words (0.25em) . This is default
  - Length
    - Defines an additional space between words (in px, pt, cm, em, etc). Negative values are allowed
  - Initial
  - Inherit

Colors

- Colors in CSS are most often specified by:
  - a valid color name - like "red"
  - an RGB value - like "rgb(255, 0, 0)"
  - a HEX value - like "#ff0000"

Background

- CSS background properties are used to define the background effects for elements.
- CSS background properties:
  - background-color
  - background-image
  - background-repeat
  - background-attachment
  - background-position

background-color

- Specifies the background color of an element
- Color is most often specified by:
  - a valid color name - like "red"
  - a HEX value - like "#ff0000"
  - an RGB value - like "rgb(255,0,0)"
  - transparent - specifies that the background color should be transparent. This is default

Background-image

- Sets one or more background images for an element
- The background of an element is the total size of the element, including padding and border (but not the margin).
- By default, a background-image is placed at the top-left corner of an element, and repeated both vertically and horizontally.

Background-image

```
body {
        background-image: url(“src");
}
```

Background Image - Repeat Horizontally or Vertically

- By default, the background-image property repeats an image both horizontally and vertically.
- Some images should be repeated only horizontally or vertically, or they will look strange, like this:

Value	Description
repeat	The background image will be repeated both vertically and horizontally. This is default
repeat-x	The background image will be repeated only horizontally
repeat-y	The background image will be repeated only vertically
no-repeat	The background-image will not be repeated
initial	Sets this property to its default value.
inherit	Inherits this property from its parent element

Background-position

- The background-position property sets the starting position of a background image.
- The background image is placed according to the background-position property. If no background-position is specified, the image is always placed at the element's top left corner

Value	Description
left top left center left bottom right top right center right bottom center top center center center bottom	If you only specify one keyword, the other value will be "center"
x% y%	The first value is the horizontal position and the second value is the vertical. The top left corner is 0% 0%. The right bottom corner is 100% 100%. If you only specify one value, the other value will be 50%. . Default value is: 0% 0%
xpos ypos	The first value is the horizontal position and the second value is the vertical. The top left corner is 0 0. Units can be pixels (0px 0px) or any other CSS units. If you only specify one value, the other value will be 50%. You can mix % and positions
initial	Sets this property to its default value. Read about initial
inherit	Inherits this property from its parent element. Read about inherit

Background-attachment

- Sets whether a background image is fixed or scrolls with the rest of the page.

Value	Description
scroll	The background scrolls along with the element. This is default
fixed	The background is fixed with regard to the viewport
local	The background scrolls along with the element's contents
initial	Sets this property to its default value.
inherit	Inherits this property from its parent element.

Background-clip(css3)

- Specifies the painting area of the background.

Value	Description
border-box	Default value. The background is clipped to the border box
padding-box	The background is clipped to the padding box
content-box	The background is clipped to the content box
initial	Sets this property to its default value.
inherit	Inherits this property from its parent element.

Background-origin(css3)

- Specifies where the background image is positioned.

Value	Description
padding-box	Default value. The background image starts from the upper left corner of the padding edge
border-box	The background image starts from the upper left corner of the border
content-box	The background image starts from the upper left corner of the content
initial	Sets this property to its default value.
inherit	Inherits this property from its parent element.

Background-size(css3)

- Specifies the size of the background images.

Value	Description
auto	Default value. The background-image contains its width and height
length	Sets the width and height of the background image. The first value sets the width, the second value sets the height. If only one value is given, the second is set to "auto"
percentage	Sets the width and height of the background image in percent of the parent element. The first value sets the width, the second value sets the height. If only one value is given, the second is set to "auto"
cover	Scale the background image to be as large as possible so that the background area is completely covered by the background image. Some parts of the background image may not be in view within the background positioning area
contain	Scale the image to the largest size such that both its width and its height can fit inside the content area
initial	Sets this property to its default value.
inherit	Inherits this property from its parent element

Borders

- Border properties allow you to specify the style, width, and color of an element's border.
  - Border-styles
  - Border-color
  - Border-width
  - border-Individual sides

Border Style

- Specifies what kind of border to display.
- The following values are allowed:
  - dotted - Defines a dotted border
  - dashed - Defines a dashed border
  - solid - Defines a solid border
  - double - Defines a double border
  - groove - Defines a 3D grooved border. The effect depends on the border-color value
  - ridge - Defines a 3D ridged border. The effect depends on the border-color value
  - inset - Defines a 3D inset border. The effect depends on the border-color value
  - outset - Defines a 3D outset border. The effect depends on the border-color value
  - none - Defines no border
  - hidden - Defines a hidden border

Border-width

- Specifies the width of the four borders.
- The width can be set as a specific size
  - in px, pt, cm, em, etc .
- By using one of the three pre-defined values:
  - thin,
  - medium, or
  - thick.
- The border-width property can have from one to four values (for the top border, right border, bottom border, and the left border).

Border-color

- Used to set the color of the four borders.
- The color can be set by:
  - name - specify a color name, like "red"
  - Hex - specify a hex value, like "#ff0000"
  - RGB - specify a RGB value, like "rgb(255,0,0)"
  - transparent
- The border-color property can have from one to four values (for the top border, right border, bottom border, and the left border).
- If border-color is not set, it inherits the color of the element.

Border – Individual Sides

- It is possible to specify a different border for each side.
- If the border-style property has four values:
- **border-style: dotted solid double dashed;**
  - top border is dotted, right border is solid, bottom border is double, left border is dashed
- If the border-style property has three values:
- **border-style: dotted solid double;**
  - top border is dotted, right and left borders are solid, bottom border is double
- If the border-style property has two values:
- **border-style: dotted solid;**
  - top and bottom borders are dotted, right and left borders are solid
- If the border-style property has one value:
- **border-style: dotted;**
  - all four borders are dotted
- The border-style property is used in the example above. However, it also works with border-width and border-color.

Margins

- Used to generate space around elements.
- All the margin properties can have the following values:
  - auto - the browser calculates the margin
  - *length* - specifies a margin in px, pt, cm, etc.
  - *%* - specifies a margin in % of the width of the containing element
  - inherit - specifies that the margin should be inherited from the parent element

Padding

- Used to generate space around content.
- The padding clears an area around the content (inside the border) of an element.
- With CSS, you have full control over the padding. There are CSS properties for setting the padding for each side of an element (top, right, bottom, and left).

Padding

- CSS has properties for specifying the padding for each side of an element:
  - padding-top
  - padding-right
  - padding-bottom
  - padding-left
- All the padding properties can have the following values:
  - *length* - specifies a padding in px, pt, cm, etc.
  - *%* - specifies a padding in % of the width of the containing element
  - inherit - specifies that the padding should be inherited from the parent element

Height & Width

- The height and width properties are used to set the height and width of an element.
- The height and width can be set to
  - auto
  - be specified in *length values*, like px, cm, etc.,
  - in percent (%) of the containing block.

- **Note:** The height and width properties do not include padding, borders, or margins; they set the height/width of the area inside the padding, border, and margin of the element!

Max-width

- It is used to set maximum-width of the element.
- It overrides the value of width property.
- It is used to give maximum width after that extent and div will not get extends.

Syntax:

max-width: none | *length* | initial | inherit;

Note: Max-width is supported by all major browsers.

Min-Width

- Set the minimum width of a <p> element.
- It is used to fix the minimum width,after the given value, div will start expanding automatically.

  Syntax:

  min-width: *length* | initial | inherit;

Note: The value of the min-width property overrides both **max-width** and **width**

Min-Height

- It is used to set the minimum height of an element.
- It applies to all elements but non-replacing the inline elements, table columns, and column groups

  Syntax:

  min-height: *length* | initial | inherit

Note:CSS3 introduces the special value inital which should be preferably used but unfortunately it has no IE support

Max-height

- The **max-height** property is used to set the maximum height of an element.
- Max-height overrides height, but min-height overrides max-height.

Syntax:

max-height:none | *length* | initial | inherit;

Link

- This is used to set different properties of hyper-link using CSS.

Properties:

:link

:visited

:hover

:active

- **a:link** - signifies unvisited hyperlinks.
- **a:visited** - signifies visited hyperlinks.
- **a:hover** - signifies an element that currently has the user's mouse pointer hovering over it.
- **a:active** - signifies an element on which the user is currently clicking.

Syntax:

:link { *style properties* }

Properties:

- Remember a:hover must come after a:link and a:visited in the CSS definition.
- Also, a:active must come after a:hover in the CSS .

Example:

```
<style type="text/css">
      a:link {color: #000000}
     a:visited {color: #006600}
     a:hover {color: #FFCC00}
     a:active {color: #FF00CC}
</style>
```

Example

```
<html>
<head>
<style type="text/css">
a:link {color:#000000}
</style>
</head>
<body>
<a href="">Link</a>
 </body>
 </html>
```

Text-Decoration:

Text-Decoration is mostly used to remove underlines from links.

Background color:

Background color is used to specify background color of links.

Link Buttons:

This are used to display links as boxex/buttons by applying css properties.

Properties for link-btns:background-color,color,padding,text align,text-decoaraton,display...

Lists

- Lists are used to convey list is either bullets or numbered points.

Properties:

Marks for Ordered-Lists.

Marks for UnOrdered-Lists.

Used to set Image as list item markers.

Background colors to lists and list items.

List-style Image

- This property allows you to use a custom image for your bullet.

Syntax:

```
ul
  {
            list-style-image: url(star.svg);
  }
```

List-style-position

- Sets whether the bullets appear inside the list items, or outside them before the start of each item.
- The <ul> and <ol> elements have a top and bottom margin of 16px (1em) and a padding-left of 40px (2.5em.)

Syntax:

```
 ul {
   list-style-position: inside;
}
```

List-style

- It allows you to control the shape or style of bullet point (also known as a marker).

Properties:

- ✓ list-style-type
- ✓ list-style-position
- ✓ list-style-image

- We can also style lists with colors.
- We can apply to entire <ol> or<ul> to affect the entire list. Or we can use for <li> tag for individual list items.

Properties:

Padding

margin-left

margin-right

Background

CSS Tables

- To improve the Html tables by applying css.

Table-Borders:

For table borders in css use the Border property.

Example:

```
table, th, td {
    border: 1px solid black;
}
```

- You can set following properties of a table
- **Border-collapse:** property sets whether the table borders should be collapsed into a single border:
- B**order-spacing:** specifies the width that should appear between table cells.
- **Caption-side:** You use the *caption-side* property to control the placement of the table caption.
- **Empty-cells:**specifies whether the border should be shown if a cell is empty cells

Table-width,Height

- This are defined by using height and width properties.

Example:

```
      table {
      width: 100%;
   }

   th {
      height: 70px;
   }
```

Horizontal Alignment

- This can be used by setting the property text-align.
- BY default it sets to center.

Syntax:

```
th {
    text-align: left;
    }
```

Vertical Alignment

- This can be used by setting the property vertical-align.
- By default, the vertical alignment of the content in a table is middle for both <th> and <td> elements

```
          td {
      height: 50px;
      vertical-align: bottom;
   }
```

Hoverable Table

- Use the hover selector on <tr> to highlight table rows on mouse over.

Syntax:

```
 tr:hover
{
background-color: blue
}
```

- **Stripped tables:** For zebra-striped tables, use the nth-child() selector and add a background-color to all even or odd table rows

Example:

```
        tr:nth-child(even) {

                background-color: #f2f2f2
}
```

Display

- The **display** CSS property specifies the type of rendering box used for an element.
- This can be applied to all HTML values.
- The default value in XML is inline.
- In HTML, default display property values are taken from behaviors described in the HTML specifications or from the browser/user default style sheet.
- The display property is the most important CSS property for controlling layout.

CSS Syntax:

display: *value;*

display:
{
block | inline | inline-block | inline-table | list-item | run-in | table | table-caption | table-cell | table-column | table-column-group | table-footer-group | table-header-group | table-row | table-row-group | none | inherit
}

Display-inside:

These keywords specify the element’s inner display type.

```
/* <display-inside> values */
   display: flow;
   display: flow-root;
   display: table;
   display: flex;
   display: grid;
   display: ruby;
   display: subgrid;
```

Display-outside:

These keywords specify the element’s outer display type.

```
/* <display-outside> values */
 display: block;
display: inline;
display: run-in;
```

/*<display-outside> plus <displainside>values*/

```
 display: block flow;
display: inline table;
display: flex run-in;
```

Display list-item:

```
    /* <display-listitem> values */
display: list-item;
display: list-item block;
display: list-item inline;
display: list-item flow;
display: list-item flow-root;
display: list-item block flow;
display: list-item block flow-root;
display: flow list-item block;
```

Display Internal values:

display: table-row-group;
display: table-header-group;
display: table-footer-group;
display: table-row;
display: table-cell;
display: table-column-group;
display: table-column;
display: table-caption;
display: ruby-base;
display: ruby-text;
display: ruby-base-container;
display: ruby-text-container;

Block-level Elements

- A block-level element always starts on a new line and takes up the full width available.

Examples of block-level elements:

- <div>
- <h1> - <h6>
- <p>
- <form>
- <header>
- <footer>
- <section>

Inline Elements

- An inline element doesn’t start on a new line and only takes up as much as width as necessary.

- <span>
- <a>
- <img>

Default Display Property

- Every element has a default display value. However, you can override this.

- Changing an inline element to a block element, or vice versa, can be useful for making the page look a specific way.

-
  ```
  li{
      display : none; // default value
  }
  ```

- Setting the display property of an element only changes how the element is displayed, NOT what kind of element it is. So, an inline element with display: block; is not allowed to have other block elements inside it.

```
span{
        display : block
}
```

Hide an Element:

Hiding an element can be done by using display as none.

All the elements are hidden, the page will be displayed as if elements are not there.

If the element is in hidden mode it doesnot take any space.

```
h1.hidden
{
  display: none;
}
```

Visibility:hidden

This also hiddens the element by using visibility:hidden property.

But it takes the element space and it effecs the layout.

```
            h1.hidden
    {
            visibility: hidden;
    }
```

Overflow

- The overflow property specifies what happens if content overflows an element's box.
- For the overflow property to have an effect, the block level container must either have a bounding height (height or max-height) or have white-space set.
- There may be a situation when the content of an element might be larger than the dimensions of the box itself

- CSS overflow property allowing you to specify whether to clip content, render scroll bars or display overflow content of a block-level element.
- This property can take one of the following values:

  visible (default),

  hidden,

  scroll,

  auto.

Value	Description
visible	The default value. Content is not clipped; it will be rendered outside the element's box, and may thus overlap other content.
hidden	Content that overflows the element's box is clipped and the rest of the content will be invisible.
Scroll	The overflowing content is clipped, just like hidden, but provides a scrolling mechanism to access the overflowed content.
auto	If content overflows the element's box it will automatically provides the scrollbars to see the rest of the content, otherwise scrollbar will not appear.

- CSS3 also defines the overflow -x and overflow -y properties which allow for independent control of the vertical and horizontal clipping.

Syntax:

overflow:
    visible | hidden | scroll | auto | initial | inherit;

```
/*Content is not clipped */
 overflow: visible;
/* Content is clipped,with no scrollbars */
overflow: hidden;
 /*Content is clipped, with scrollbars */
overflow: scroll;
 /* Let the browser decide */
overflow: auto;
/* Global values */
 overflow: inherit;
overflow: initial;
 overflow: unset;
```

Other properties for overflow:

- Overflow-text
- Overflow-whitespace
- Overflow-x
- Overflow-y
- Clip-display

Text-Overflow

- This property determines how overflowed content that is not displayed is signaled to users.
- Clipping happens at the border of the box.
- This CSS property doesn't force an overflow to occur; to do so and make text-overflow to be applied.

Syntax:

```
/* Overflow behavior at line end Right end if ltr,
left end if rtl */
 text-overflow: clip;
text-overflow: ellipsis;
text-overflow: "…";

/* Overflow behavior at left end | at right end
Directionality has no influence */
 text-overflow: clip ellipsis;
text-overflow: "…" "…";
```

Whitespace

- The **white-space** property is used to describe how whitespace inside the element is handled.

  Syntax:

```
/* Keyword values */
 white-space: normal;
white-space: nowrap; white-space: pre;
white-space: pre-wrap;
white-space: pre-line;
/* Global values */
 white-space: inherit;
white-space: initial;
white-space: unset;
```

Values for white-space

Normal:

Sequences of whitespace are collapsed. Breaks lines as necessary to fill line boxes.

No-wrap:

Collapses whitespace as for normal, line breaks (text wrapping) within text.

Pre:

Sequences of whitespace are preserved. Lines are only broken at newline characters in the source and at <br> elements.

Pre-line:

Sequences of whitespace are collapsed. Lines are broken at newline characters, at <br>, and as necessary to fill line boxes.

overflow-x and overflow-y

- Properties specifies whether to change the overflow of content just horizontally or vertically or both.

  Overflow-x ---> left /right

  Overflow-y---> top/bottom

Example:

```
div {
        /* Hide horizontal scrollbar */
      overflow-x: hidden;
   /* Add vertical scrollbar */
       overflow-y: scroll;
}
```

Float

- The float property allow to implement simple layouts involving an image floating inside a column of text, with the text wrapping around the left or right of it.
- The kind of thing you might get in a newspaper layout.
- How Elements Float: A floated element is taken out of the normal flow and shifted to the left or right as far as possible in the space available of the containing element.

- If several floating elements are placed adjacently, they will float next to each other if there is horizontal room. If there is not enough room for the float, it is shifted downward until either it fits or there are no more floating elements present.
- Value ---> Description
- left ---> The element floats on the left side of its containing block.
- right ---> The element floats on the right side of its containing block.
- none ---> Removes the float property from an element.
- Note:This property can clear an element only from floated boxes within the same block. It doesn't clear the element from floated child boxes within the element itself.

Clearfix

- The clear property is used to control the behavior of floating elements. It specifies which sides of an element's box other floating elements are not allowed.
- clearfix can take three values:
  - **left**: Stop any active left floats
  - **right**: Stop any active right floats
  - **both**: Stop any active left and right floats

The clearfix Hack:

- If an element is taller than the element containing it, and it is floated, it will overflow outside of its container.
- Then we can add overflow: auto; to the containing element to fix this problem.
- The overflow: auto clearfix works well as long as you are able to keep control of your margins and padding (else you might see scrollbars).

- clearfix hack however, is safer to use, and the following code is used for most webpages.

Eg:

```
.clearfix::after
   {
     content: “ ";
     clear: both;
     display: table;
   }
```

Inline-block

- Inline-block elements are like inline elements but they can have a width and a height.
- It has been possible for a long time to create a grid of boxes that fills the browser width and wraps nicely when the browser is resized, by using the float property.
- However, the inline-block value of the display property makes this even easier.

Syntax

```
.floating-box
 {
   display: inline-block;
   width: 150px;
   height: 75px;
}
```

Alignment

- CSS has several properties that can be used to align elements on the web pages:
  - Text Alignment.
  - Center Alignment Using the margin Property.
  - Aligning Elements Using the position Property.
  - Left and Right Alignment Using the float Property.
  - Clearing Floats

Text Alignment:

Text inside the block-level elements can aligned by setting the text-align properly.

Example:

```
.center
{
text-align: center;
border: 3px solid green;
}
```

Center Alignment Using the margin

- Center alignment of a block-level element is one of the most important implications of the CSS margin property.
- . For example, the <DIV> container can be aligned horizontally center by setting the left and right margins to 'auto'.

Note:The value auto for the margin property will not work in Internet Explorer 8 and earlier versions, unless a <DOCTYPE > is specified

Right/Left align Using

- When aligning elements with float, always define margin and padding for the <body> element.
- This is to avoid visual differences in different browsers.
- For aligning elements we use float property.

Example:

```
.right
{
    float: right;
    width: 300px;
    border: 3px solid #73AD21;
    padding: 10px;
}
```

Selectors

- In CSS, selectors are patterns used to select the elements you want to style.
- CSS selectors are used to "find" (or select) HTML elements based on their element name, id, class, attribute, and more.
  - Element
  - Id
  - Class
  - Grouping

Element Selector

- The element selector selects elements based on the element name.
- You can select all <p> elements on a page like this (in this case, all <p> elements will be center-aligned, with a red text color):

Syntax:

```
element
{
css declarations


}
```

Id selector

- The id selector uses the id attribute of an HTML element to select a specific element.
- The id of an element should be 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.
- The style rule below will be applied to the HTML element with id="para1":

Syntax:

```
    #id
{
    css declarations;
  }
```

Example:

```
#firstname
{
    background-color: yellow;
    font-size:10px;
  }
```

Class selector

- The class selector selects elements with a specific class attribute.
- To select elements with a specific class, write a period (.) character, followed by the name of the class.

Syntax:

```
.class
  {
        css declarations;
  }
```

Grouping selector

- It will be better to group the selectors, to minimize the code.
- To group selectors, separate each selector with a comma.

Syntax:

```
td, th
{
    : declarations
}
```

Combinators

- A selector can contain more than one simple selector. Between the simple selectors, we must include a combinator
- It explains the relationship between the selectors.
- There are three different combinators in CSS2, and one extra in CSS3.
- When we use them, they change the nature of the selector to reflect.

The following are combinators for selectors:

- Adjacent sibiling selectors
- General sibiling selectors
- Child selectors
- Descedent selectors

Adjacent Sibiling selector:

- This is referred to as an adjacent selector or next-sibling selector.
- It will select only the specified element that immediately follows the former specified element.

Syntax:

```
        former_element + target_element
 {
 style properties
}
```

General sibling selectors

- The ~ combinator separates two selectors and matches the second element only if it is preceded by the first, and both share a common parent.

Syntax:

```
    element ~ element
 {
     style properties
 }
```

Child selectors

- The > combinator separates two selectors and matches only those elements matched by the second selector that are **direct** children of elements matched by the first.

Syntax:

```
    selector1 > selector2
{
     style properties
 }
```

Descendant selectors

- typically represented by a single space ( ) character — combines two selectors such that elements matched by the second selector are selected if they have an ancestor element matching the first selector.
- The descendant combinator is technically one or more CSS white space characters

Syntax:

selector1 selector2 { / property declarations */ }*

selector1 >> selector2 { / property declarations */ }*

Pseudo-classes

- A CSS ***pseudo-class*** is a keyword added to selectors that specifies a special state of the element to be selected.
- A CSS pseudo-class selector matches components based on an additional condition and not necessarily defined by the document tree.
- The CSS pseudo-classes allow you to style the dynamic states of an element such as hover, active and focus state, as well as elements that are existing in the document tree but can't be targeted via the use of other selectors without adding any IDs or classes to them

- For example, targeting the first or last child elements.
- A pseudo-class starts with a colon (:).

Syntax:

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

Simply with a colon in between the selector and the pseudo class.

The most commonly used pseudo-classes.

Anchor Pseudo-classes:

Using anchor pseudo-classes links can be displayed in different ways-

- These pseudo-classes let you style unvisited links differently from visited ones. The most common styling technique is to remove underlines from visited links.
- Some anchor pseudo-classes are dynamic — they're applied as a result of user interaction with the document like on hover, or on focus etc.

- These pseudo-classes change how the links are rendered in response to user actions:
  - :**hover** applies when a user places cursor over the element, but does not select it.
  - :**active** applies when the element is activated or clicked.
  - :**focus** applies when the element has keyboard focus.

Note : To make these pseudo-classes work perfectly, you must define them in the exact order – :link, :visited, :hover, :active, :focus.

The :first-child Pseudo-class:

- The :first-child pseudo-class matches an element that is the first child element of some other element.
- The selector 'ol li:first-child' in the example below select the first list item of an ordered list and removes the top border form it.

Note:To make :first-child to work in Internet Explorer 8 and earlier versions, a <Doctype > must be declared at the top of document.

The :last-child Pseudo-class:

- The :last-child pseudo-class matches an element that is the last child element of some other element.
- The selector 'ul li:last-child' in the example below select the last list item from an unordered list and removes the right border from it.

Note : The CSS :last-child selector does not work in Internet Explorer 8 and earlier versions. Supports in Internet Explorer 9 and above

The :nth-child Pseudo-class:

- The CSS3 introduces a new :nth-child pseudo-class that allows you to target one or more specific children of a given parent element.
- The basic syntax of this selector can be given with :nth - child(N), where N is an argument, which can be a number, a keyword (even or odd), or an expression of the form xn+y where x and y are integers (e.g. 1n, 2n, 2n+1, …).
- The style rules in the example above simply highlight the alternate table row, without adding any IDs or classes to the table elements.

The : lang Pseudo-class

- The language pseudo-class :lang allows constructing selectors based on the language setting for specific tags.
- The :lang pseudo-class in the example below defines the quotation marks for q elements that are explicitly given a language value of "no".

Note : Internet Explorer up to version 7 does not support the : lang pseudo-class. IE8 supports only if a Doctype is specified.

Pseudo-classes and CSS Classes

- Pseudo-classes can be combined with CSS classes.
- The link with class="red", in the example below will be displayed in red.

CSS Pseudo-elements

- The CSS pseudo-elements is a ways to style elements of the document that weren't explicitly defined by a position in the document tree.

What is Pseudo-element ??

The CSS pseudo-elements allow you to style the elements or parts of the elements without adding any IDs or classes to them.

It will be really helpful in the situations when you just want to style the first letter of a paragraph to create the drop cap effect or you want to insert some content before or after an element, etc. only through style sheet.

- CSS3 introduced a new double-colon (::) syntax for pseudo-elements to distinguish between them and pseudo-classes. The new syntax of the pseudo-element can be given with:

Syntax:

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

- These are the following most commonly used pseudo-elements:

The :: first-line Pseudo-element—

The ::first-line pseudo-element applies special style to the first line of a text.

The style rules in the following example formats the first line of text in a paragraph.

The length of first line depends on the size of the browser window or containing element.

The ::first-letter Pseudo-element:

The ::first-letter pseudo-element is used to add a special style to the first letter of the first line of a text.

The style rules in the following example formats the first letter of the paragraph of text and create the effect like drop cap.

Example:

```
        p::first-letter
    {
       color: #ff0000;
       font-size: xx-large;
    }
```

The :: before and ::after Pseudo-element:

- The ::before and ::after pseudo-elements can be used to insert generated content either before or after an element's content.
- The content CSS property is used in conjunction with these pseudo-elements, to insert the generated content.
- This is very useful for further decorating an element with rich content that should not be part of the page's actual markup.
- You can insert regular strings of text or an embedded object such as image and other resources using these pseudo-elements.

Pseudo-elements and CSS Classes:

- Generally we need to style only a certain paragraph of text or other block-level elements with these pseudo-elements.
- That's where declaring a class to the pseudo element comes into play.
- The pseudo-elements can be combined with the CSS Classes to produce the effect particularly for the elements having that class.

Opacity

- The opacity CSS property specifies the transparency of an element.

Cross Browser Opacity:

Opacity is now a part of the CSS3 specifications, but it was present for a long time.

However, older browsers have different ways of controlling the opacity or transparency.

CSS Opacity in Firefox, Safari, Chrome, Opera and IE9

Here is the most up to date syntax for CSS opacity in all current browsers.

CSS Opacity in Internet Explorer 8 and lower:

Internet Explorer 8 and earlier version supports a Microsoft-only property "alpha filter" to control the transparency of an element.

Note:Alpha filters in IE accept values from 0 to 100. The value 0 makes the element completely transparent (i.e. 100% transparent), whereas the value 100 makes the element completely opaque (i.e. 0% transparent).

CSS Opacity for All Browser:

Combining the both steps above you will get the opacity for all browsers.

Note : Including alpha filter to control transparency in Internet Explorer 8 and lower versions creates invalid code in your style sheet since this is a Microsoft-only property, not a standard CSS property.

CSS Image Opacity:

You can also make transparent images using CSS Opacity.

The three images in the illustration below are all from the same source image.

The only differences between them are the level of their opacity.

Exapmle:

```
img {
    opacity: 0.5;
    filter: alpha(opacity=50);
  /* For IE8 and earlier */
  }
```

Change Image Opacity on Mouse Over:

where the opacity of images changes when the user moves the mouse pointer over an image.

Example:

```
 img {
 opacity: 0.5;
filter: alpha(opacity=50); /* For IE8 and earlier */
}

img:hover {
 opacity: 1.0;
 filter: alpha(opacity=100); /* For IE8 and earlier
*/
}
```

Text in Transparent Box:

- When using opacity on an element not only the background of the element that will have transparency, but all of its child elements become transparent as well.
- It is making the text inside the transparent element hard to read if the value of opacity becomes higher.

Example:

```
div {
   opacity: 0.3;
   filter: alpha(opacity=30);
}
```

CSS Transparency Using RGBA:

In addition to RGB CSS3 has introduced a new way RGBA to specify a color that includes alpha transparency as part of the color value. RGBA stands for Red Blue Green Alpha.

The RGBA declaration is a very easy way to set transparency for a color.

Note : The RGBA transparency doesn't affect the child elements the way the opacityproperty does. The alpha value of RGBA affects the transparency of individual color rather than the entire element.

Declaring a Fallback Color:

All browsers do not support RGBA colors. However, you can provide an alternative such as solid colors or transparent PNG images for the browsers that don't support it.

Note : Internet Explorer 8 and earlier versions do not support the RGBA colors. They use the gradient-filter to achieve the effect of RGBA, which is deprecated

Navigation Bars

- It has several special classes to display a navigation bar or a menu on a website quickly.

Class Name	Description
Top-nav	Styles a list as a horizontal menu/navigation bar.
Side-nav	Styles a list as a verical menu/navigation bar.

Navigation Bar = List of Links

A navigation bar needs standard HTML as a base.

In our examples we will build the navigation bar from a standard HTML list.

A navigation bar is basically a list of links, so using the <ul> and <li> elements makes perfect sense:

Example:

```
<ul>
    <li><a href="default.asp">Home</a></li>
    <li><a href="news.asp">News</a></li>
    <li><a href="contact.asp">Contact</a></li>
    <li><a href="about.asp">About</a></li>
  </ul>
```

Vertical Navigation Bar:

To build a vertical navigation bar, you can style the <a> elements inside the list.

Example:

```
    li a {
    display: block;
    width: 60px;
 }
```

- Create a basic vertical navigation bar with a gray background color and change the background color of the links when the user moves the mouse over them

Example:

```
* Change the link color on hover */
li a:hover {
    background-color: #555;
    color: white;
}
```

Active/Current Navigation Link

- Add an "active" class to the current link to let the user know which page he/she is on:

  Example:

```
    .active
 {
    background-color: #4CAF50;
    color: white;
}
```

Center Links & Add Borders

- Add text-align:center to <li> or <a> to center the links.
- Add the border property to <ul> add a border around the navbar. If you also want borders inside the navbar, add a border-bottom to all <li> elements, except for the last one

Example:

```
li {
    text-align: center;
    border-bottom: 1px solid #555;
}

li:last-child {
    border-bottom: none;
}
```

Full-height Fixed Vertical Navbar

- Create a full-height, "sticky" side navigation

  Example:

```
ul {
     margin: 0;
    padding: 0;
    width: 25%;
    background-color: #f1f1f1;
    height: 100%;
    position: fixed;
    overflow: auto;
}
```

Horizontal Navigation Bar

- There are two ways to create a horizontal navigation bar. Using **inline** or **floating** list items.

Inline List Items:

One way to build a horizontal navigation bar is to specify the <li> elements as inline, in addition to the "standard”.

Example:

```
li {
  display: inline;
  }
```

Floating List Items:

Another way of creating a horizontal navigation bar is to float the <li> elements, and specify a layout for the navigation links.

Example:

```
    li {


    float: left;
}

a {
    display: block;
    padding: 8px;
    background-color: #dddddd;
}
```

Active/Current Navigation Link:

Add an "active" class to the current link to let the user know which page the user is.

Example:

```
    .active
{
    background-color: #4CAF50;
 }
```

Dropdowns

- Create a dropdown box that appears when the user moves the mouse over an element.
- A not-uncommon flavor of navigation is the **dropdown** menu, where sub-navigation lists only appear when the cursor passes over a link.
- The :hover selector is used to show the dropdown menu when the user moves the mouse over the dropdown button.

Dropdown Menu:

User can select the option from the dropdown.

Except that we add links inside the dropdown box and style them to fit a styled dropdown button.

Right-aligned Dropdown Content:

If we want place the dropdown menu to go from right to left, instead of left to right.

Example:

```
.dropdown-content {
   right: 0;
}
```

Forms

- Forms are the standard way to receive user inputted data. The transitions and smoothness of these elements are very important because of the inherent user interaction associated with forms.

Input fields:

Text fields allow user input. The border should light up simply and clearly indicating which field the user is currently editing

- If you only want to style a specific input type, you can use attribute selectors:

- input[type=text] - will only select text fields
- input[type=password] - will only select password fields
- input[type=number] - will only select number fields
- etc..

Textarea

- Textareas allow larger expandable user input. The border should light up simply and clearly indicating which field the user is currently editing.
- You must have a .input-field div wrapping your input and label.

Textareas will auto resize to the text inside:
When dynamically changing the value of a textarea with methods like jQuery's .val(), you must trigger an autoresize on it afterwords because .val() does not automatically trigger the events we've binded to the textarea.

Select:
Select allows user input through specified options.
Make sure you wrap it in a .input-field for proper alignment with other text fields.
Remember that this is a jQuery plugin so make sure you initialize this in your document ready.

File Input

- If you want to style an input button with a path input we provide this structure.

Example:

```
    <form action="#">
    <div class="file-field input-field">
     <div class="btn">
     <span>File</span>
     <input type="file">
     </div> <div class="file-path-wrapper">
    <input class="file-path validate" type="text">
    </div> </div>
</form>
```

Display-Position

- The *position* property is used in positioni ng an element.
- It can be used with the top, right, bottom and left properties to position an element where you want it.
- The **position** CSS property chooses alternative rules for positioning elements.
- Mainly designed to be useful for scripted animation effects.

Properties

- The properties used for display:position are as follows
    - static
    - relative
    - fixed
    - absolute

/* Global values */

position: inherit;

position: initial;

position: unset;

Static

- This keyword lets the element use the normal behavior.
- Elements are positioned static by default.
- The top, right, bottom, left and z-index properties do not apply.(static position elements)

Syntax

```
    div.static
{
    position: static;
    border: 3px solid #73AD21;
}
```

Relative

- This keyword lays out all elements as though the element were not positioned, and then adjusts the element's position.
- Without changing layout and thus leaving a gap for the element.
- The effect of position: relative on

  table-*-group, table-row, table-column, table-cell, and table-caption elements is undefined.

Syntax

```
div.Relative
        {
          position: relative;
          left: 20px;
          border: 3px solid #73AD21;
        }
```

Fixed

- It always stays in the same place even if the page is scrolled.
- The top, right, bottom, and left properties are used to position the element.
- Do not leave space for the element.
- The element’s box is absolutely positioned, with all of the behaviors which are described for position: absolute.
- The major difference is that the containing block of a fixed-position element is always the viewport.

Syntax

```
    div.fixed
{
    position: fixed;
    bottom: 0;
    right: 0;
    width: 300px;
    border: 5px solid #73AD21;
  }
```

Absolute

- It do not leave space for the element.
- Positioned boxes can have margins, and they do not collapse with any other margins.
- It moves along with the page scrolling.
- Also, an absolutely positioned element can have margins, and they do not collapse with any other margins.

Syntax

```
img
{
      position: absolute;
      left: 0px;
      top: 0px;
      z-index: -1;
  }
```

Tooltips

- A tooltip is often used to specify extra information about something when the user moves the mouse pointer over an element.
- Moving the mouse over top of a toolbar button and waiting for a second can display a small popup label containing text describing the button's function. When the mouse is moved again, the tooltip disappears.
- This tooltips can only be activated using the mouse, so they should never contain important information that isn't available elsewhere.

Position:

- Bottom
- Top
- Left
- Right

Example:

```
  /* Position the tooltip */
position: absolute;
z-index: 1;
top: -10px;
right: 105%;a
```

Top Tooltip:

```
.tooltip .tooltiptext
  {
 width: 120px;
  bottom: 100%;
  left: 50%;
  margin-left: -60px;
  }
```

Tooltip Arrows

- Create an arrow that should appear from a specific side of the tooltip, add "empty" content after tooltip.
- With the pseudo-element class ::after together with the content property.
- The arrow itself is created using borders.
- This will make the tooltip look like a speech bubble.

Fade

- Fade in the tooltip text when it is about to be visible.
- This can be used by the property “Opacity”.

Example:

```
    .tooltip .tooltiptext
 {
       opacity: 0;
       transition: opacity 1s;
    }

    .tooltip:hover .tooltiptext {
       opacity: 1;
    }
```

Image gallery

- Images play an important role in any webpage.
- CSS plays a good role to control image display.

Properties:

border

height

width

opacity

Image Border

- The *border* property of an image is used to set the width of an image border.
- This property can have a value in length or in%.
- A width of zero pixels means no border.

Sprites

- CSS sprites technique is a way to reduce the number of HTTP requests made for image resources, by combining images in a single file.
- Sprites are two-dimensional images which are made up of combining small images into one larger image at defined X and Y coordinates.
- To display a single image from the combined image, you could use the CSS background-position property, defining the exact position of the image to be displayed.

Advantage of Using CSS Image Sprite:

A web page with many images, particularly many small images, such as icons, buttons, etc. can take a long time to load and generates multiple server requests.

Using the image sprites instead of separate images will significantly reduce the number of HTTP requests a browser makes to the server, which can be very effective for improving the loading time of web pages and overall site performance.

Display an Icon from Image Sprite:

- Utilizing CSS, we can display just the part of an image sprite we need.
- the class 'sprite' that will load our sprite image. This is to avoid repetition, because all items share the same background-image.
- Whereas in sprite version, since all images are combined in a single image the hover image is displayed immediately on mouse hover that results in smooth hover effect.

Hover effect

- The :hover selector can be used on all elements, not only on links.
- There will be **no loading delay** when a user hovers over the image.

Example:

```
#home a:hover {
    background: url('img_navsprites_hover.gif') 0 -45px;
}

#prev a:hover {
    background: url('img_navsprites_hover.gif') -47px -45px;
}

#next a:hover {
    background: url('img_navsprites_hover.gif') -91px -45px;
}
```