Quadrant IV – Module Assessment

Programme: Bachelor of Arts / Bachelor of Science

Subject: (GE)

Paper Code: CSG110

Paper Title: CLIENT SIDE WEB DEVELOPMENT

Unit: Unit III: Introduction to CSS

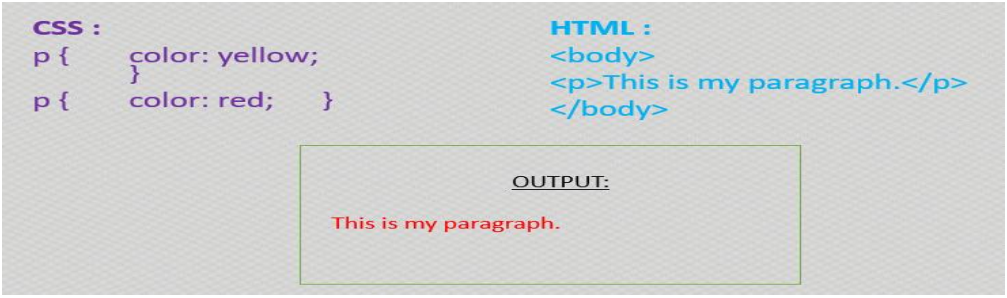
Module Name: Cascade and inheritance, box model, fundamental text and font styling, values, units, colors, media queries, layout- static, liquid, adaptive and responsive, floats, positioning, flex box, grids.

Module No: 17

Name of the Presenter: Ms. Vandana . O. Sawant
Assistant Professor
Government College of Arts, Science and Commerce,
Khandola-Marcela, Goa

The cascade

- In CSS, the order in which we specify our rules matters.
- If a rule from the same style sheet, with the same level of specificity exists, the rule that is declared last in the CSS document will be the one that is applied.



Inheritance

- Inheritance is the mechanism by which certain properties are passed on from a parent element down to its children.
- In CSS, some styles are **inherited** down the HTML document tree while others are not.

- Not all CSS properties are inherited.

For instance, margins and width are not inherited, since it is unlikely that a child element requires the same margins as its parent.

- CSS has an inheritance mechanism because otherwise CSS rules would be **redundant**.

Without inheritance, it would be necessary to specify styles like font family, font size, and text color individually — for every single element type. The code would become bloated and repetitive.

- Every element in an HTML document inherits all inheritable properties from its parent except the root element (<html>), which does not have a parent.
- Whether or not the inherited properties will have any effect depends on the cascade and specificity.

Example:

- The body element in our HTML is the parent of all of our other HTML elements (excluding the <head> section).
- Setting the font property on the body element allows the rest of the document to inherit the font rule.

```
body { font: 14px/18px Helvetica, Verdana, sans-serif; }
```

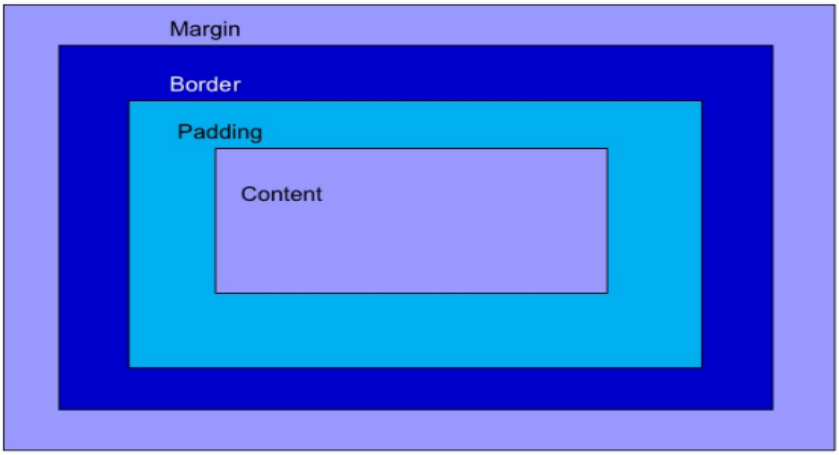
- Using the rule above, all text (unless we specify otherwise) will be 14px with an 18px line height and be Helvetica or Verdana.

CSS Box Model

- Defined as a tool used to lay out our web pages in a number of individual "**boxes**" or "**containers**".
- HTML elements can be considered as boxes.
- In CSS, the term "box model" is used when referring to layout.

WHY BOX Model ?

The CSS Box Model is used to create a definition for the way the HTML elements are organized on the screen. This approach accounts for options such as margins, padding, borders, and all the properties that manipulate them.
Each element can be thought of as having its own box.
Components of a box model



- **The Margin**

On the CSS Box Model, the margin is the section in the exterior. This edge extends around the box model taking the empty space between the margin and border properties. You can consider this a buffer area that separates the interior of the CSS box model from other HTML elements on a page.

- **The Border**

Next, you have the Border property of the CSS Box Model. Keeping in the theme of separation, this area exists as a boundary between the margin and padding properties of the box. This area can be manipulated using the CSS Border property for styling and sizing needs.

- **The Padding**

The padding section of the CSS Box Model sits in the space between the HTML content and the border. As with much of the other items in the box mode, the padding can be altered through its related properties. For example, the padding can be changed through the directions of the top, right, bottom, and left sections.

- **The Content**

This is the main HTML content you are working to display on the site. This can range from an image, a paragraph, or even a web button. While this is quite direct in definition, it should be noted that “content” can be empty space as well. Using an empty div for example can be a neat way to add extra creative components to a site design.

Fundamental text and font styling

The CSS properties used to style text generally fall into two categories:

Font styles: Properties that affect a text's font, e.g., which font gets applied, its size, and whether it's bold, italic, etc.

Text layout styles: Properties that affect the spacing and other layout features of the text, allowing manipulation of, for example, the space between lines and letters, and how the text is aligned within the content box.

CSS Fonts

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

- **Color**
  The color property sets the color of the foreground content of the selected elements, which is usually the text, but can also include a couple of other things, such as an underline or overline placed on text using the text-decoration property.

  Example:
  p {color: red;}

- **Font families**
  to set a different font for your text, you use the font-family property. This allows you to specify a font (or list of fonts) for the browser to apply to the selected elements.

  Example:
  p {font-family: arial;}

- **Font size**
  Font size can take values measured in most of these units (and others, such as percentages); however, the most common units you'll use to size text are:
  - **px (pixels):** The number of pixels high you want the text to be.
  - **ems:** 1 em is equal to the font size set on the parent element of the current element we are styling (more specifically, the width of a capital letter M contained inside the parent element).
  - **rems:** These work just like em, except that 1 rem is equal to the font size set on the root element of the document (i.e. <html>), not the parent element.

- **Font style**
  Used to turn italic text on or off. Possible values are as follows
  - **normal:** Sets the text to the normal font
  - **italic:** Sets the text to use the italic version of the font, if available; if not, it will simulate italics with oblique instead.
  - **oblique:** Sets the text to use a simulated version of an italic font, created by slanting the normal version.

- **Font weight**
  Sets how bold the text is. This has many values available in case you have many font variants available (such as -light, -normal, -bold, -extrabold, -black, etc.), but realistically you'll rarely use any of them except for normal and bold.

- **Text transform**
  Allows you to set your font to be transformed. Values include:
  - **none:** Prevents any transformation.
  - **uppercase:** Transforms all text to capitals.
  - **lowercase:** Transforms all text to lower case.
  - **capitalize:** Transforms all words to have the first letter capitalized.
  - **full-width:** Transforms all glyphs to be written inside a fixed-width square, similar to a monospace font, allowing aligning of, e.g., Latin characters along with Asian language glyphs (like Chinese, Japanese, Korean).

- **Text decoration**
  Sets/unsets text decorations on fonts (you'll mainly use this to unset the default underline on links when styling them). Available values are:
  - **none:** Unsets any text decorations already present.
  - **underline:** Underlines the text.
  - **overline:** Gives the text an overline.
  - **line-through:** Puts a strikethrough over the text.

- **Text Drop Shadows**

  You can apply drop shadows to your text using the text-shadow property. This takes up to four values, as shown in the example below:

  text-shadow: h1 {text-shadow: 2px 2px 5px red;}

  The four properties are as follows:

  - The horizontal offset of the shadow from the original text — this can take most available CSS length and size units, but you'll most commonly use px; positive values move the shadow right, and negative values left. This value has to be included.
  - The vertical offset of the shadow from the original text.
  - The blur radius: a higher value means the shadow is dispersed more widely.
  - The base color of the shadow, which can take any CSS color unit.

Text layout

Let's have a look at properties we can use to affect text layout.

- **Text alignment**

  The text-align property is used to control how text is aligned within its containing content box. The available values are listed below.

  - **left:** Left-justifies the text.

- **right:** Right-justifies the text.
- **center:** Centers the text.

❑ **Line height**

The line-height property is used to specify the space between lines:

Example :

```
p.small {
  line-height: 0.7;
}
p.big {
  line-height: 1.8;
}
```

This is a paragraph with a standard line-height.
The default line height in most browsers is about 110% to 120%.

This is a paragraph with a smaller line-height.
This is a paragraph with a smaller line-height.

This is a paragraph with a bigger line-height.

This is a paragraph with a bigger line-height.

❑ **Letter spacing**

The letter-spacing property is used to specify the space between the characters in a text.

The following example demonstrates how to increase or decrease the space between characters:

```
h2 {
letter-spacing: 5px;
}
h3 {
letter-spacing: -2px;
}
```

This is heading 1

This is heading 2

❑ **Word spacing**

The word-spacing property is used to specify the space between the words in a text.
The following example demonstrates how to increase or decrease the space between words:

```
p.one {
```

```
  word-spacing: 10px;
}
p.two {
  word-spacing: -2px;
}
```

This is a paragraph with normal word spacing.

This is a paragraph with larger word spacing.

This is a paragraph with smaller word spacing.

CSS Units

- For some CSS properties, negative lengths are allowed.
- There are two types of length units: absolute and relative.

❑ Absolute Lengths: The absolute length units are fixed and a length expressed in any of these will appear as exactly that size.

- Absolute length units are not recommended for use on screen, because screen sizes vary so much. However, they can be used if the output medium is known, such as for print layout.
- An adaptive layout uses fixed units such as pixels, just like a static layout.
- What distinguishes an adaptive layout from a static layout is its use of media queries.
- By using media queries, an adaptive layout changes when the viewport is at certain widths.
- Device is served the layout with closet possible match.

❑ Relative Lengths: Relative length units specify a length relative to another length property. Relative length units scale better between different rendering medium.

- A responsive layout uses relative units such as percentages and ems, just like a liquid layout.
- It takes the best from both liquid and adaptive layouts, using both relative units and media queries. This makes it possible to deliver the right layout for each device and viewport size.
- If you want to give your users a great experience no matter what device they are using, using a responsive layout is the way to go.

CSS Colors

Colors in CSS can be specified by the following methods:

✓ Hexadecimal colors

✓ Hexadecimal colors with transparency

- ✓ RGB colors
- ✓ RGBA colors
- ✓ HSL colors
- ✓ HSLA colors
- ✓ Predefined/Cross-browser color names
- ✓ With the currentcolor keyword

CSS Media Queries

- Media queries in CSS3 extended the CSS2 media types idea: Instead of looking for a type of device, they look at the capability of the device.

  Media queries can be used to check many things, such as:

  - ❑ width and height of the viewport
  - ❑ width and height of the device
  - ❑ orientation (is the tablet/phone in landscape or portrait mode?)
  - ❑ resolution

CSS Layout

When talking about layouts in CSS, we often divide the different types of layouts into 4 categories:

1) Static (also called a fixed layout)

   In a static layout, the size of elements are set using fixed units, such as pixels.

2) Liquid (also called a fluid layout)

   A problem with liquid layouts is that some elements may become too narrow or too wide. This can cause the layout to break, and make the text harder to read.

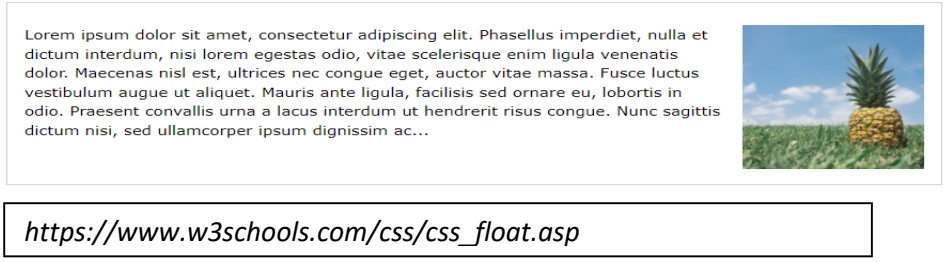
3) Adaptive

4) Responsive

CSS Layout – float property

- The CSS float property specifies how an element should float.
- The float property is used for positioning and formatting content
- The float property can have one of the following values:

- ❑ left
- ❑ Right
- ❑ none
- ❑ inherit



https://www.w3schools.com/css/css_float.asp

CSS Layout - The position Property

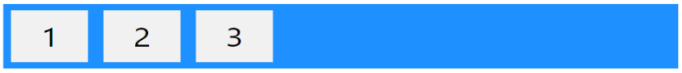
- The position property specifies the type of positioning method used for an element.
- There are five different position values:
  - ❑ static
  - ❑ relative
  - ❑ fixed
  - ❑ absolute
  - ❑ Sticky
- Elements are then positioned using the top, bottom, left, and right properties.
- However, these properties will not work unless the position property is set first. They also work differently depending on the position value

CSS Flexbox

- Before the Flexbox Layout module, there were four layout modes:
  - ❑ Block, for sections in a webpage
  - ❑ Inline, for text
  - ❑ Table, for two-dimensional table data
  - ❑ Positioned, for explicit position of an element
- The Flexible Box Layout Module, makes it easier to design flexible responsive layout structure without using float or positioning.

❑ **Flexbox Elements**

- To start using the Flexbox model, you need to first define a flex container.
- The element above represents a flex container (the blue area) with three flex items.
- Example
  - A flex container with three flex items:
  -
  - 1
  - 2
  - 3
  -



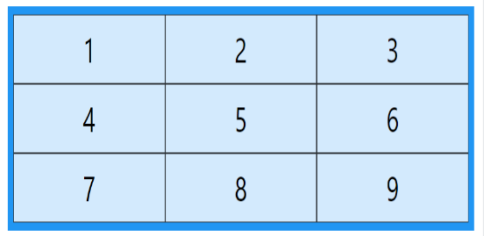
CSS-Grid

- The CSS Grid Layout Module offers a grid-based layout system, with rows and columns, making it easier to design web pages without having to use floats and positioning.
- Grid Elements

A grid layout consists of a parent element, with one or more child elements.

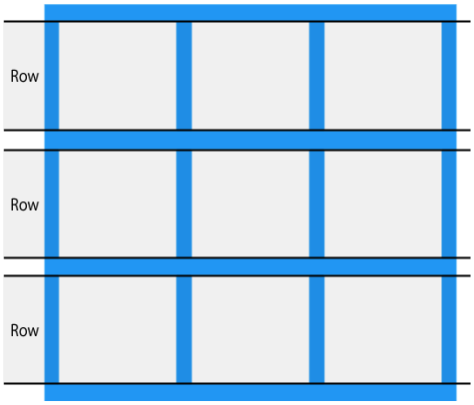
- Example

```
<div class="grid-container">
  <div class="grid-item">1</div>
  <div class="grid-item">2</div>
  <div class="grid-item">3</div>
  <div class="grid-item">4</div>
  <div class="grid-item">5</div>
  <div class="grid-item">6</div>
  <div class="grid-item">7</div>
  <div class="grid-item">8</div>
  <div class="grid-item">9</div> </div>
```



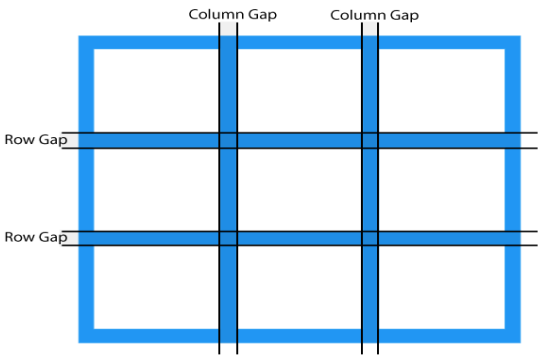
Grid Rows

- The horizontal lines of grid items are called *rows*.



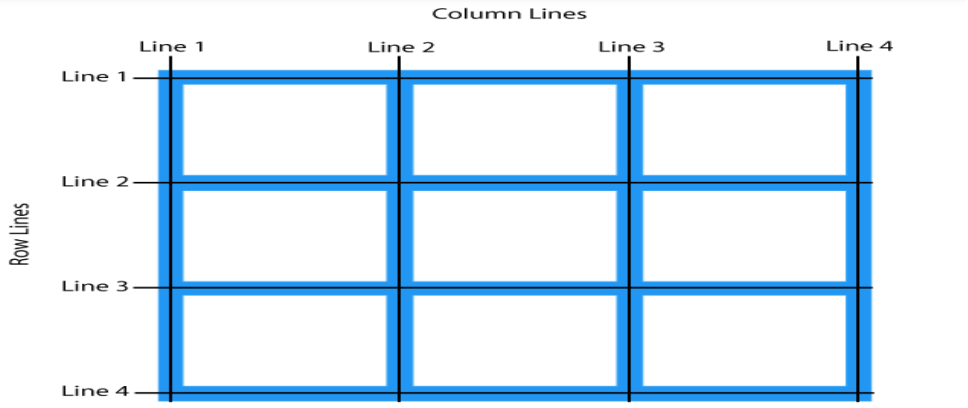
Grid Gaps

- The spaces between each column/row are called *gaps*.
- You can adjust the gap size by using one of the following properties:
  - column-gap
  - row-gap
  - gap



Grid Lines

- The lines between columns are called *column lines*.
- The lines between rows are called *row lines*.
- Refer to line numbers when placing a grid item in a grid container:



Grid Lines Example

Place a grid item at column line 1, and let it end on column line 3:

```
.item1 {
  grid-column-start: 1;
  grid-column-end: 3;
}
```