Cascading Style Sheets (CSS)

Introduction to CSS

- CSS stands for Cascading Style Sheets
- 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
- CSS can be added to HTML elements in 3 ways:
  - **Inline** - by using the **style attribute** in HTML elements
  - **Internal** - by using a **<style> element** in the **<head> section**
  - **External** - by using an external CSS file
- External stylesheets are stored in CSS files

Example

Inline CSS

```
<h1 style="color:blue;">This is a Blue Heading</h1>
```

Internal CSS

```
<!DOCTYPE html>
<html>
    <head>
        <style>
            h1   {color: blue;}
            p    {color: red;}
        </style>
    </head>
    <body>
        <h1>This is a heading</h1>
        <p>This is a paragraph.</p>
    </body>
</html>
```

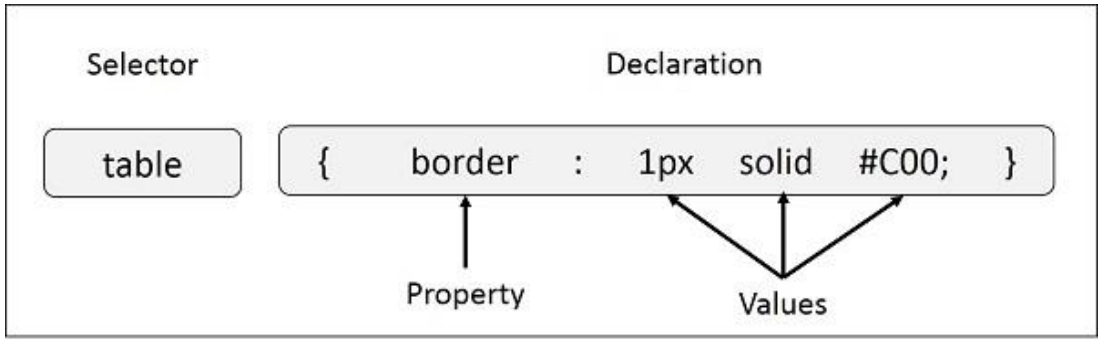
HTML Section(test.html)

```
<!DOCTYPE html>
<html>
<head>
        <link rel="stylesheet" type="text/css"
href="styles.css">
</head>
<body>

        <h1>This is a heading</h1>
        <p>This is a paragraph.</p>

</body>
</html>
```

CSS Section (styles.css)

```
body {
  background-color: yellow;
}
h1 {
  color: blue;
}
p {
  color: red;
}
```

Syntax



- A CSS rule-set consists of a selector and a declaration block:
- **Selector** – A selector is an HTML tag at which a style will be applied. This could be any tag like <h1> or <table> etc.
- **Property** – A property is a type of attribute of HTML tag. Put simply, all the HTML attributes are converted into CSS properties. They could be color, border etc.
- **Value** – Values are assigned to properties. For example, color property can have value either black or #000000 etc.

CSS Selectors

- CSS selectors are used to "find" (or select) the HTML elements you want to style.
- We can divide CSS selectors into five categories:
  - Simple selectors: select elements based on name, id, class
  - Combinator selectors: select elements based on a specific relationship between them
  - Pseudo-class selectors: select elements based on a certain state
  - Pseudo-elements selectors: select and style a part of an element
  - Attribute selectors: select elements based on an attribute or attribute value

The CSS element Selector

- The element selector selects HTML elements based on the element name.
- Example

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

The CSS id Selector

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

```
#para1 {
  text-align: center;
  color: red;
}
```

Note: An id name cannot start with a number!

The CSS class Selector

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

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

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

```
<p class="center">This paragraph refers to two classes.</p>
```

Note: A class name cannot start from number!

Example

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

```
p.large {
  font-size: 300%;
}
```

```
<h1 class="center">Will not be affected</h1>
<p class="center">Will be red and center-aligned.</p>
<p class="center large">center-aligned, and in a large font-size.</p>
```

The CSS Universal Selector

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

```
* {
  text-align: center;
  color: blue;
}
```

The CSS Grouping Selector

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

Without Grouping

```
h1 {
  text-align: center;
  color: red;
}

h2 {
  text-align: center;
  color: red;
}

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

With Grouping

```
h1, h2, p {
  text-align: center;
  color: red;
}
```

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 CSS:
  - descendant selector (space)
  - child selector (>)
  - adjacent sibling selector (+)
  - general sibling selector (~)

Descendant Selector

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

```
<!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>
 <section>
      <p>Paragraph 3 in the div.</p>
</section>
<p>Paragraph 4. Not in a div.</p>
<p>Paragraph 5. Not in a div.</p>

</body>
</html>
```

Child Selector

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

```
<!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>
  <section><p>Paragraph 3 in the
div.</p></section>
<p>Paragraph 4 in the div.</p>
</div>
<p>Paragraph 5. Not in a div.</p>
<p>Paragraph 6. Not in a div.</p>
</body>
</html>
```

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:

```
<!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>
```

General Sibling Selector

- 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:

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

</body>
</html>
```

CSS Pseudo-classes

- A pseudo-class is used to define a special state of an element.
- For example, it can be used to:
  - Style an element when a user mouse over it
  - Style visited and unvisited links differently
  - Style an element when it gets focus

Example

```
<!DOCTYPE html>
<html>
<head>
</head>
<body>
<p><b><a href="default.asp"
target="_blank">This is a
link</a></b></p>
</body>
</html>
```

```
<style>
/* unvisited link */
a:link {
  color: red;
}
/* visited link */
a:visited {
  color: green;
}
/* mouse over link */
a:hover {
  color: hotpink;
}
/* selected link */
a:active {
  color: blue;
}
</style>
```

CSS Backgrounds

CSS background-color

- The background-color property specifies the background color of an element.
- With CSS, a 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)“

Example:

```
body {
  background-
color: lightblue;
}
```

CSS background-image and background-repeat

- The background-image property specifies an image to use as the background of an element.
- By default, the image is repeated so it covers the entire element.
- By default, the background-image property repeats an image both horizontally and vertically.
- To control the repeating we use background-repeat so the background will look better
(Values: repeat-x, repeat-y, no- repeat)

```
body {
  background-image: url("gradient_bg.png");
  background-repeat: repeat-x;
}
```

CSS background-position background-attachment

- The background-position property is used to specify the position of the background image.
- The background-attachment property specifies whether the background image should scroll or be fixed.

```
body
{
  background-image: url("img_tree.png");
  background-repeat: no-repeat;
  background-position: right top;
  background-attachment: fixed;
}
```

CSS background - Shorthand property

- It is also possible to specify all the background properties in one single property. This is called a shorthand property.
- The shorthand property for background is background.

```
body {
  background: #ffffff url("img_tree.png") no-repeat right top;
}
```

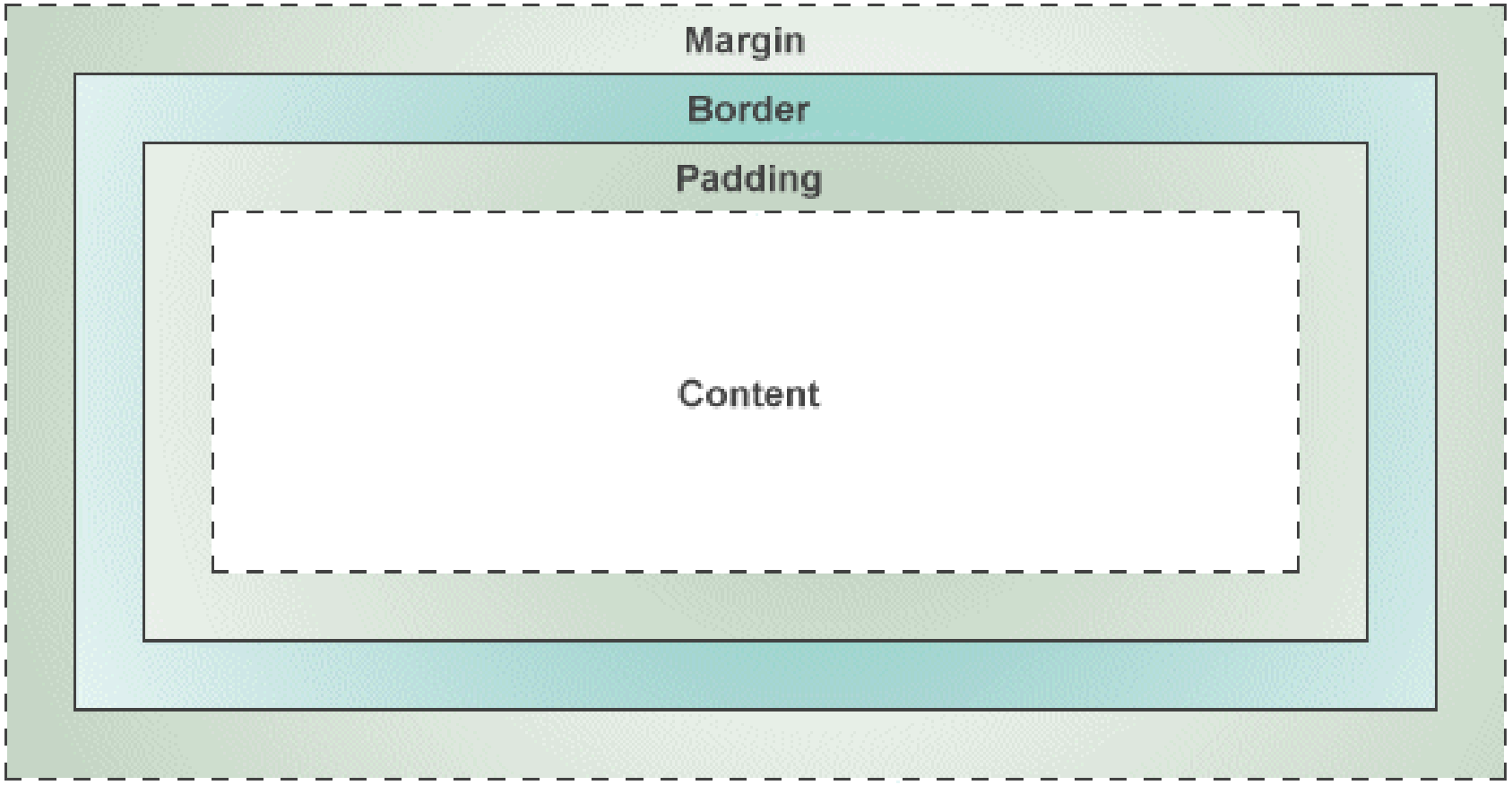
  - background-color
  - background-image
  - background-repeat
  - background-attachment
  - background-position

Pseudo-classes and CSS Classes

- Pseudo-classes can be combined with CSS classes:
- When you hover over the link in the example, it will change color:

```
<!DOCTYPE html>
<html>
<head>
<style>
a.highlight:hover {
  color: #ff0000;
}
</style>
</head>
<body>

<p><a class="highlight" href="css_syntax.asp">CSS Syntax</a></p>
<p><a href="default.asp">CSS Tutorial</a></p>

</body>
</html>
```

Box Model



Box model is the way of representing the element/content in a web page.

Cont..

- **Content** - The content of the box, where text and images appear
- **Padding** - Clears an area around the content. The padding is transparent
- **Border** - A border that goes around the padding and content
- **Margin** - Clears an area outside the border. The margin is transparent

Source: https://www.w3schools.com/css/css_boxmodel.asp

CSS Margins

Margin

- The CSS margin properties are used to create space around elements, outside of any defined borders.
- CSS has properties for specifying the margin for each side of an element:
  - margin-top
  - margin-right
  - margin-bottom
  - margin-left
- 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

Margin - Shorthand Property

- The margin property is a shorthand property for the following individual margin properties:
  - margin-top
  - margin-right
  - margin-bottom
  - margin-left

```
p {
  margin: 25px 50px 75px 100px;
}
```

  - top margin is 25px
  - right margin is 50px
  - bottom margin is 75px
  - left margin is 100px

Cont..

- p {
  margin: 25px 50px 75px;
  }
- top margin is 25px
- right and left margins are 50px
- bottom margin is 75px

Cont..

- p {
  margin: 25px 50px;
  }
- top and bottom margins are 25px
- right and left margins are 50px

CSS Borders

CSS Border Style

- The border-style property specifies what kind of border to display.
- The border-style property can have from one to four values (for the top border, right border, bottom border, and the left border).
- 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

CSS Border Width and Border Color

- The border-width property specifies the width of the four borders.
- The width can be set as a specific size (in px, pt, cm, em, etc)
- The border-width property can have from one to four values (for the top border, right border, bottom border, and the left border).
- The border-color property is used to set the color of the four borders.

```
p.one {
  border-style: solid;
  border-width: 5px;
  border-color: red;
}
p.three {
  border-style: solid;
  border-width: 2px 10px 4px 20px;
}
```

CSS Border - Individual Sides

- In CSS, there are also properties for specifying each of the borders (top, right, bottom, and left):

```
p {
  border-top-style: dotted;
  border-right-style: solid;
  border-bottom-style: dotted;
  border-left-style: solid;
}
```

CSS Border - Shorthand Property

- The border property is a shorthand property for the following individual border properties:
  - border-width
  - border-style (required)
  - border-color

```
p {
  border: 5px solid red;
}
```

CSS Rounded Borders

- The border-radius property is used to add rounded borders to an element

```
p {
  border: 2px solid red;
  border-radius: 5px;
}
```

CSS Padding

CSS Padding

- The CSS padding properties are used to generate space around an element's content, inside of any defined borders.
- Padding
- padding-top
- padding-right
- padding-bottom
- padding-left
- Ex:

```
div {
 padding-top: 50px;
 padding-right: 30px;
 padding-bottom: 50px;
 padding-left: 80px;
}
```

CSS height and width

- The height and width properties are used to set the height and width of an element.
- The height and width properties may have the following values:
  - auto - This is default. The browser calculates the height and width
  - length - Defines the height/width in px, cm etc.
  - % - Defines the height/width in percent of the containing block
  - initial - Sets the height/width to its default value
  - inherit - The height/width will be inherited from its parent value

Setting max-width

- The max-width property is used to set the maximum width of an element.
- The max-width can be specified in length values, like px, cm, etc., or in percent (%) of the containing block, or set to none (this is default. Means that there is no maximum width).

```
div {
  width:1000px;
  max-width: 500px;
  height: 100px;
  background-color: powderblue;
}
```

- The problem with the above occurs when the browser window is smaller than the width of the element (500px). The browser then adds a horizontal scrollbar to the page.

CSS | Text Formatting

Operations	Properties	Values
Text-color	color	• a color name - "red" • a HEX value - "#ff0000" • an RGB value - "rgb(255,0,0)"
Text-alignment	text-align	center, left, right, justify
Text-decoration	text-decoration	overline, line-through, underline, None
Text-transformation	text-transform	uppercase, lowercase, capitalize
Text-indentation	text-indent	Number value (indentation of the first line of a text)
Letter spacing	letter-spacing	Number value
Line height	line-height	Number value (Space between the lines)
Text-direction	direction	rtl, ltr
Text-shadow	text-shadow	Number value
Word spacing	word-spacing	Number value

CSS Fonts

- The CSS font properties define the font family, boldness, size, and the style of a text.
- Since not all fonts are available on all computers, CSS provides a system of fallbacks.
- CSS Font Families
- In CSS, there are five font family names:
  - serif,
  - sans-serif,
  - monospace,
  - cursive
  - fantasy.

Font Family

- The font family of a text is set with the font-family property.
- The font-family property should hold several font names as a "fallback" system. If the browser does not support the first font, it tries the next font, and so on.
- Start with the font you want, and end with a generic family, to let the browser pick a similar font in the generic family, if no other fonts are available.
- Example:

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

Font Style

- The font-style property is mostly used to specify italic text.
- This property has three values:
  - normal - The text is shown normally
  - italic - The text is shown in italics
  - oblique - The text is "leaning" (oblique is very similar to italic, but less supported)

Font Size

- The font-size property sets the size of the text.
- Being able to manage the text size is important in web design. However, you should not use font size adjustments to make paragraphs look like headings, or headings look like paragraphs.
- Always use the proper HTML tags, like <h1> - <h6> for headings and <p> for paragraphs.
- 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

Example

```
h1 {
  font-size: 40px;
}

h2 {
  font-size: 30px;
}

p {
  font-size: 14px;
}
```

```
h1 {
  font-size: 2.5em; /* 40px/16=2.5em */
}
h2 {
  font-size: 1.875em; /* 30px/16=1.875em
*/
}
p {
  font-size: 0.875em; /* 14px/16=0.875em
*/
}
```

Font Weight

- The font-weight property specifies the weight of a font
- Example

```
p.normal {
  font-weight: normal;
}

p.thick {
  font-weight: bold;
}
```

Responsive Font Size

- The text size can be set with a vw unit, which means the "viewport width".
- Example

```
<h1 style="font-size:10vw">Hello World</h1>
```

Font Variant

- The font-variant property specifies whether or not a text should be displayed in a small-caps font.
- Example

```
p.normal {
  font-variant: normal;
}

p.small {
  font-variant: small-caps;
}
```

Float

- The float CSS property places an element on the left or right side of its container, allowing text and inline elements to wrap around it.
- The CSS float property specifies how an element should float.
  - left - The element floats to the left of its container
  - right- The element floats to the right of its container
  - none - The element does not float (will be displayed just where it occurs in the text). This is default
  - inherit - The element inherits the float value of its parent

The position Property

- The position property specifies the type of positioning method used for an element
  - **Static** - this is the default value, all elements are in order as they appear in the document.
  - **Relative** - the element is positioned relative to its normal position.
  - **Absolute** - the element is positioned absolutely to its first positioned parent.
  - **Fixed** - the element is positioned related to the browser window.
  - **Sticky** - the element is positioned based on the user's scroll position.

Display

- Every HTML element has a default display value depending on what type of element it is. The default display value for most elements is block or inline.
- The display property specifies if/how an element is displayed.
  - Block
  - Inline
  - Block-inline

Block-level Elements

- A block-level element always starts on a new line and takes up the full width available (stretches out to the left and right as far as it can).
- Examples of block-level elements:
  -
  - <h1> - <h6>
  - <p>
  - <form>
  - <header>
  - <footer>
  - <section>

Inline Elements

- An inline element does not start on a new line and only takes up as much width as necessary.
- Examples of inline elements:
  -
  - <a>
  - <img>

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.

Display: none;

- display: none; is commonly used with JavaScript to hide and show elements without deleting and recreating them. Take a look at our last example on this page if you want to know how this can be achieved.
- The <script> element uses display: none; as default.

Overflow

- The CSS overflow property controls what happens to content that is too big to fit into an area.
- Visible: The overflow is not clipped. It renders outside the element's box. This is default
- Hidden: The overflow is clipped, and the rest of the content will be invisible
- Scroll: The overflow is clipped, but a scroll-bar is added to see the rest of the content
- Auto: If overflow is clipped, a scroll-bar should be added to see the rest of the content

Listing

- list-style: Sets all the properties for a list in one declaration
- list-style-image: Specifies an image as the list-item marker
- list-style-position: Specifies the position of the list-item markers (bullet points)
- list-style-type: Specifies the type of list-item marker

CSS Layout - Horizontal & Vertical Align

Responsive Web Design

What is Responsive Web Design?

- Responsive web design makes your web page look good on all devices.
- Responsive web design uses only HTML and CSS.
- Responsive web design is not a program or a JavaScript.

The Viewport

- The viewport is the user's visible area of a web page.
- The viewport varies with the device, and will be smaller on a mobile phone than on a computer screen.
- Before tablets and mobile phones, web pages were designed only for computer screens, and it was common for web pages to have a static design and a fixed size.
- Then, when we started surfing the internet using tablets and mobile phones, fixed size web pages were too large to fit the viewport. To fix this, browsers on those devices scaled down the entire web page to fit the screen.

Setting The Viewport

- HTML5 introduced a method to let web designers take control over the viewport, through the <meta> tag.
- *<meta name="viewport" content="width=device-width, initial-scale=1.0">*
- The *width=device-width* part sets the width of the page to follow the screen-width of the device (which will vary depending on the device).
- The *initial-scale=1.0* part sets the initial zoom level when the page is first loaded by the browser.

Grid-View

- Many web pages are based on a grid-view, which means that the page is divided into columns
- A responsive grid-view often has 12 columns, and has a total width of 100%, and will shrink and expand as you resize the browser window.

CSS Media Queries

CSS2 Media Types

- The @media rule, introduced in CSS2, made it possible to define different style rules for different media types.
- Examples: You could have one set of style rules for computer screens, one for printers, one for handheld devices, one for television-type devices, and so on.
- Unfortunately these media types never got a lot of support by devices, other than the print media type.

CSS3 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
- Using media queries are a popular technique for delivering a tailored style sheet to desktops, laptops, tablets, and mobile phones.

Media Query Syntax

- A media query consists of a media type and can contain one or more expressions, which resolve to either true or false.

```
@media not|only mediatype and (expressions)
{
  CSS-Code;
}
```

```
<link rel="stylesheet" media="mediatype and|not|only (expressions)" href="print.css">
```

CSS3 Media Types

Value	Description
all	Used for all media type devices
print	Used for printers
screen	Used for computer screens, tablets, smart-phones etc.
speech	Used for screen readers that "reads" the page out loud

```
@media screen and (min-width: 480px)
{
  body {
    background-color: lightgreen;
  }
}
```

Table Borders and Collapse Table Borders

Border

- To specify table borders in CSS, use the border property.
- The example below specifies a black border for <table>, <th>, and <td> elements:

```
table {
  border-collapse: collapse;
}

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

Border Collapse

- The border-collapse property sets whether the table borders should be collapsed into a single border

Table Width and Height

- Width and height of a table are defined by the width and height properties.
- The example below sets the width of the table to 100%, and the height of the <th> elements to 50px:

```
table {
  width: 100%;
}

th {
  height: 50px;
}
```

Horizontal and Vertical Alignment

- The text-align property sets the horizontal alignment (like left, right, or center) of the content in <th> or <td>.
- The vertical-align property sets the vertical alignment (like top, bottom, or middle) of the content in <th> or <td>.

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

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

Table Padding

- To control the space between the border and the content in a table, use the padding property on <td> and <th> elements

z-index property

- The z-index property specifies the stack order of an element.
- An element with greater stack order is always in front of an element with a lower stack order.
- Syntax: z-index: auto|*number*|initial|inherit;

Note: z-index only works on positioned elements (position: absolute, position: relative, position: fixed, or position: sticky).

Table Borders and Collapse Table Borders

Border

- To specify table borders in CSS, use the border property.
- The example below specifies a black border for <table>, <th>, and <td> elements:

```
table {
  border-collapse: collapse;
}

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

Border Collapse

- The border-collapse property sets whether the table borders should be collapsed into a single border

Table Width and Height

- Width and height of a table are defined by the width and height properties.
- The example below sets the width of the table to 100%, and the height of the <th> elements to 50px:

```
table {
  width: 100%;
}

th {
  height: 50px;
}
```

Horizontal and Vertical Alignment

- The text-align property sets the horizontal alignment (like left, right, or center) of the content in <th> or <td>.
- The vertical-align property sets the vertical alignment (like top, bottom, or middle) of the content in <th> or <td>.

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

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

Table Padding

- To control the space between the border and the content in a table, use the padding property on <td> and <th> elements

CSS Flexbox

CSS Flexbox Layout Module

- 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 flex container becomes flexible by setting the display property to flex
- The flex container properties are:
  - flex-direction- defines in which direction the container wants to stack the flex items (column, column-reverse, row, row-reverse).
  - flex-wrap - specifies whether the flex items should wrap or not (wrap, row-wrap, wrap-reverse).
  - flex-flow - is a shorthand property for setting both the flex-direction and flex-wrap properties.
  - justify-content - used to align the flex items (center, flex-start, flex-end, space-around, space-between)
  - align-items - used to align the flex items vertically (center, flex-start, flex-end)
  - align-content - used to align the flex lines (space-around, space-between, stretch, center, flex-start, flex-end, stretch, baseline)

CSS flex Property

Definition and Usage

- The flex property sets the flexible length on flexible items.
- The flex property is a shorthand property for:
  - flex-grow - A number specifying how much the item will grow relative to the rest of the flexible items
  - flex-shrink - A number specifying how much the item will shrink relative to the rest of the flexible items
  - flex-basis - The length of the item. Legal values: "auto", "inherit", or a number followed by "%", "px", "em" or any other length unit

CSS Syntax

flex: flex-grow flex-shrink flex-basis|auto|initial|inherit;

Pseudo-Elements

What are Pseudo-Elements?

- A CSS pseudo-element is used to style specified parts of an element.
- For example, it can be used to:
  - Style the first letter, or line, of an element
  - Insert content before, or after, the content of an element
- Syntax:

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

CSS Attribute Selectors

CSS Attribute Selectors

- The [attribute] selector is used to select elements with a specified attribute.

```
a[target] {
  background-color: yellow;
}
```

Selector	Example	Example description
[attribute]	[target]	Selects all elements with a target attribute
[attribute=value]	[target=_blank]	Selects all elements with target="_blank"
[attribute~=value]	[title~=flower]	Selects all elements with a title attribute containing the word "flower"
[attribute|=value]	[lang|=en]	Selects all elements with a lang attribute value starting with "en"
[attribute^=value]	a[href^="https"]	Selects every <a> element whose href attribute value begins with "https"
[attribute$=value]	a[href$=".pdf"]	Selects every <a> element whose href attribute value ends with ".pdf"
[attribute*=value]	a[href*="manipal"]	Selects every <a> element whose href attribute value contains the substring "w3schools"