Chapter 18:
Box Model in CSS

18.1 - Understanding the Box Model:

In CSS, the box model is a crucial concept for design and layout. It treats every HTML element as a box, whether it's a paragraph, image, or heading. The box model allows you to control the positioning and arrangement of these boxes in relation to each other. You achieve this by managing the margins, borders, padding, and actual content of each "box."

When working with the box model, it's vital to calculate the total size of the elements accurately. When you set the width and height of an element in CSS, you're only defining the content's size. To determine the total size, you must consider the content, padding, borders, and margins. The formula for calculating the total width is: Total width = width of content + left padding + right padding + left margin + right margin + left border + right border. The same concept applies to height, with "left" and "right" replaced by "top" and "bottom." It's worth noting that each side needs individual calculation, as each part can be considered a separate component.

Example:

p.one {width: 220px; padding: 10px; border: 5px solid maroon; margin: 0px;}

Result:

Sets the paragraph element with the ID "one" to have a total width of 250 pixels and a 5-pixel solid maroon border.

18.2 - Managing Margins:

In CSS, the MARGIN property controls the space around an element outside of the assigned border. You can manage the margin as a whole or control each side individually using properties like MARGIN-LEFT, MARGIN-RIGHT, MARGIN-TOP, MARGIN-BOTTOM, or MARGIN. You can specify margin values as "auto" (letting the browser define the size), a percentage (%), or using units like pixels (px), points (pt), centimeters (cm), millimeters (mm), inches (in), picas (pc), x-height (ex), or em. Importantly, the margin of HTML elements is always transparent and isn't influenced by any background color applied to the element. When using individual control properties, you must provide a value; otherwise, it defaults to 0px.

Property:

margin, margin-left, margin-right, margin-top, margin-bottom

Value:

auto, a percentage (%), or px, pt, cm, mm, in, pc, ex, or em

Example:

p.one {margin: 50px;}

Result:

Defines the CSS rule for paragraph "one" with a 50-pixel margin on all sides.

18.3 - Adjusting Padding:

The PADDING property in CSS determines the space within an element's border and between its content. Padding is affected by the element's background color and takes on the same color. You can express padding values as percentages (%) or in units such as pixels (px), points (pt), centimeters (cm), millimeters (mm), inches (in), picas (pc), x-height (ex), or em. Similar to margin properties, padding can be controlled individually for each side using properties like PADDING-LEFT, PADDING-RIGHT, PADDING-TOP, PADDING-BOTTOM, or PADDING. If you use individual control properties, you must assign a value to each property to avoid the default value of 0px.

Property:

padding, padding-left, padding-right, padding-top, padding-bottom

Value:

a percentage (%), or px, pt, cm, mm, in, pc, ex, or em

Example:

p.one {padding-left: 5px; padding-right: 5px; padding-top: 10px; padding-bottom: 10px;}

Result:

Establishes the CSS rule for paragraph "one" with 5-pixel padding on both the left and right sides and 10-pixel padding on both the top and bottom.

18.4 - Controlling Borders:

To set borders for an element in CSS, you use the BORDER-STYLE property before adjusting border size and color. BORDER-STYLE offers values like solid, double, groove, ridge, inset, outset, dotted, or dashed. You can control border size using BORDER-WIDTH, specifying values like thin, medium, thick, or units such as pixels (px), points (pt), centimeters (cm), millimeters (mm), inches (in), picas (pc), x-height (ex), or em. Border color is managed through the BORDER-COLOR property, accepting color names, hexadecimal values, or "transparent." Individual sides of the border can be customized using properties like BORDER-LEFT-STYLE, BORDER-RIGHT-STYLE, BORDER-TOP-STYLE, and BORDER-BOTTOM-STYLE.

Property: border-style

Value: solid, double, groove, ridge, inset, outset, dotted, dashed, or none

Property: border-width

Value: thin, medium, thick, or units like px, pt, cm, mm, in, pc, ex, or em

Property: border-color

Value: color name, hexadecimal value, or transparent

Example: p.one {border-style: dashed; border-width: medium; border-color: red;}

Result: Defines the CSS rule for paragraph "one" with a medium-sized, dashed border in the color red.

18.5 - Adding Outlines:

The outline property in CSS is used to emphasize an element on an HTML page. It's drawn outside the element's border and doesn't affect the element's total size. To define an outline style, you use the OUTLINE-STYLE property with values like solid, double, groove, ridge, inset, outset, dotted, dashed, or none. The width of the outline can be controlled using OUTLINE-WIDTH, specifying values like thin, medium, thick, or units such as pixels (px), points (pt), centimeters (cm), millimeters (mm), inches (in), picas (pc), x-height (ex), or em. Outline color is determined by the OUTLINE-COLOR property, accepting color names or hexadecimal values.

Property: outline-style

Value: solid, double, groove, ridge, inset, outset, dotted, dashed, or none

Property: outline-width

Value: thin, medium, thick, or units like px, pt, cm, mm, in, pc, ex, or em

Property: outline-color

Value: color name or hexadecimal value

Example: p.one {outline-style: double; outline-width: thin; outline-color: #FF0000;}

Result: Establishes the CSS rule for paragraph "one" with a thin, double outline in red.

Chapter 19 Working with CSS Elements

19.1 - Display and Visibility:

There are two methods for controlling the visibility of elements on a webpage: the DISPLAY property and the VISIBILITY property. Although they serve similar functions, they yield different outcomes in terms of page layout.

The DISPLAY property allows you to either hide an element or set it as a block or inline element. The core values for the DISPLAY property are: none (hides the element completely), block (displays the element with line breaks before and after the content), and inline (displays the element without line breaks). Examples of block elements include <p>, , and <h1>, while inline elements include <a> and .

Using the VISIBILITY property to hide an element still reserves space for it in the layout, resulting in a blank space where the element would be. The VISIBILITY property values are: visible (default), hidden, and collapse. The collapse value is useful for quickly removing rows or columns from a table without recalculating the table's overall dimensions.

Property: display

Value: none, block, inline

Property: visibility

Value: visible, hidden, collapse

Example: h1.hidden {display: none;}

h3.hide {visibility: hidden;}

Result: Hides all h1 headers with the "hidden" class, leaving no space, and hides all h3 headers with the "hide" class, preserving blank space on the page.

19.2 - Grouping and Nesting:

Grouping selectors is advantageous when you want to alter the appearance of multiple elements simultaneously. For instance, you can group elements like headers and paragraphs and apply CSS rules collectively. You can also style selectors within other selectors.

Grouping Example: h1, p {color: purple;}

Result: Sets all level 1 headings and paragraphs to have a purple color.

Nesting Example: p {color: black; text-align: left;} .definitions {background-color: blue;} .definitions p {color: #FFFFFF;}

Result: Makes all paragraphs have black text and left alignment, elements with class "definitions" have a blue background, and paragraphs within "definitions" have white text.

19.3 - Managing Dimensions:

You can control the size of elements using height and width properties, as well as set maximum and minimum dimensions with max-height, max-width, min-height, and min-width attributes. Elements without specific dimensions expand horizontally to the browser window's width and only occupy the content's size vertically. These attributes can be expressed in pixels, percentages, or as "auto" to let the browser calculate the size.

Attributes: height, width, min-height, min-width, max-height, max-width

Values: pixels, percentage, auto

Example: p.one {width: 100px; max-height: 100px; background-color: blue;}

Result:

Sets paragraphs with class "one" to a width of 100 pixels, a maximum height of 100 pixels, and a blue background.

19.4 - Positioning:

Element positioning on a webpage can be controlled using four position properties: STATIC (default), FIXED, RELATIVE, and ABSOLUTE. These properties are adjusted with top, bottom, left, and right attributes using pixel values. Positioning attributes won't function without a position property assignment and work differently based on the chosen property.

STATIC is the default property, FIXED keeps an element fixed relative to the browser window, RELATIVE moves an element relative to its normal position, and ABSOLUTE positions an element as if it doesn't affect page flow. The Z-INDEX property controls stacking order, with lower values behind higher ones.

Properties: fixed, relative, absolute

Values: top, bottom, right, left (numerical pixels)

Examples: p.first {position: fixed; top: 5px; right: 100px;}

h1.first {position: relative; top: -10px; right: -10px;}

h2 {position: absolute; left: 100px; top: 150px;}

Result: Defines the positions of elements, with specific properties and offsets.

19.5 - Floating:

The FLOAT property positions an element to the left or right of another element, allowing subsequent elements to flow around it. Values for FLOAT are "left" or "right." Elements appearing before the floated one remain unaffected.

Property: float

Values: left or right

Example: ul {float: left; width: 100%; padding: 1; margin: 1; list-style-type: none;}

a {float: left; width: 6em; text-decoration: none; color: black; background-color: yellow; padding: 4px; border-right: 2px solid blue;}

a:hover {background-color: #C0C0C0;}

li {display: inline;}

Result: Creates a horizontal list of links with specified styles.

19.6 - Pseudo-Classes/Pseudo-Elements:

CSS pseudo-classes and pseudo-elements provide a means to customize specific elements within your webpage. For instance, you can style the first line of a paragraph differently in terms of font, color, or emphasis compared to the rest of the paragraph using the :first-line pseudo-class/pseudo-element.

The syntax for utilizing CSS pseudo-classes and pseudo-elements is structured as follows: selector:pseudo-class {property: value;}. You can also combine a CSS class with a pseudo-class by including the class in the selector: selector.class:pseudo-class {property: value;}. There are currently eleven distinct CSS pseudo-classes available for applying special effects to your webpage.

It's important to note that CSS pseudo-classes and pseudo-elements are not case-sensitive. When employing these selectors, it's crucial to remember that there should be no space between the selector, colon, and pseudo-class/pseudo-element.

Here are examples of some commonly used CSS pseudo-classes and their functions:

- :link (e.g., a:link): Selects all unvisited links.

- :visited (e.g., a:visited): Selects all visited links.

- :hover (e.g., a:hover): Selects links on mouseover (a:hover must come after a:link and a:visited to take effect).

- :active (e.g., a:active): Selects the currently active link (a:active must come after a:hover to take effect).

- :focus (e.g., input:focus): Selects the input element currently in use.

- :first-letter (e.g., p:first-letter): Selects the first letter of every paragraph element.

- :first-line (e.g., p:first-line): Selects the first line of every paragraph element.

- :first-child (e.g., p:first-child): Selects every paragraph element that is the first child of its parent element.

- :before (e.g., p:before): Sets a rule to insert content before every paragraph element.

- :after (e.g., p:after): Sets a rule to insert content after every paragraph element.

- :lang (e.g., p:lang(was)): Selects every paragraph element with a lang="was" attribute.

19.6- Pseudo-Classes/Pseudo-Elements (cont.):

Examples:

p:first-child i {color: blue;}

a:visited {color: #FF00FF;}

p.one:first-line {font-weight: 500;}

Result: Sets CSS rules to style specific elements based on pseudo-classes or pseudo-elements.

Chapter 20: Adding a Navigation Bar in CSS

20.1 - Creating a Vertical Navigation Bar:

A navigation bar essentially functions as a list of links. To initiate the construction of your navigation bar, you must begin with a fundamental list of links. This list should be established in your HTML code, and you can then apply styling using CSS.

To form a basic vertical navigation bar, the process involves eliminating bullets, margins, and padding from the list of links using the LIST-STYLE-TYPE property set to "none." Set the margins and padding to zero, employ the DISPLAY property with a "block" value, and specify a width. This configuration yields a vertical navigation bar. You can further customize the appearance of your block element as desired. By setting the DISPLAY property to "block," the entire area becomes clickable, not just the text of the link. This means you can click anywhere within the block to activate the link.

Remember that when employing a vertical navigation bar, it's important to define a width; otherwise, the list will extend across the entire width of the browser window.

Example:

ul.vertnav {list-style-type: none; margin: 0; padding: 0;} a {display: block; width: 80px; background-color: #FFFFFF;}

Result:

A vertical navigation bar that is 80 pixels wide with a white background, utilizing the list with the class "vertnav."

20.2 - Creating a Horizontal Navigation Bar - Inline:

There are two methods for crafting a horizontal navigation bar. The first and simplest approach involves displaying your list of links with a DISPLAY property set to "inline" for your list items (li). This arranges your list to display horizontally side by side. The LIST-STYLE-TYPE property is set in the same way as for a vertical navigation bar, removing list markers and setting margin and padding to zero.

Example:

ul.horiznav {list-style-type: none; margin: 0; padding: 0;} li {display: inline;}

Result:

A horizontal navigation bar utilizing the list with the class "horiznav."

20.3 - Creating a Horizontal Navigation Bar - Floating:

The second approach to establish a horizontal navigation bar involves using the FLOAT property to position the links in a horizontal row. When employing the FLOAT property, it is essential to assign it to float the individual list items (li) rather than the entire unordered list (ul). This distinction is significant, as you will need to specify a CLASS for each list item.

Example:

ul {list-style-type: none; margin: 0; padding: 0;} li.horiznav {float: left;} a {display: block; width: 60px;}

Result:

A horizontal navigation bar where the list of links with the class "horiznav" is floated from left to right and displayed as a block element with a width of 60 pixels.