Web Intruders

Lecture no 7

Margin:

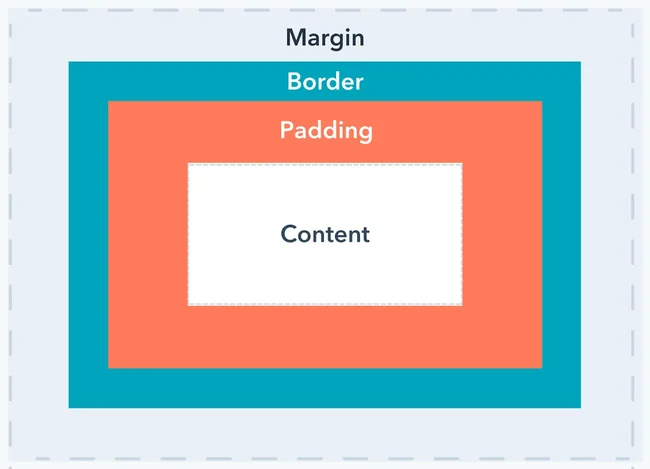
Margins create space around an element. They are the space outside the border, creating separation between the element and surrounding elements. You can set margins using properties like **margin-top**, **margin-right**, **margin-bottom**, and **margin-left**.

Border:

Borders surround the content and padding of an element. They can be styled, colored, and sized using properties like **border-width**, **border-style**, and **border-color**. Borders can also have rounded corners using **border-radius**.

**Padding**:

Padding is the space between the content of an element and its border. It provides internal spacing within an element. Padding can be set using properties like **padding-top**, **padding-right**, **padding-bottom**, and **padding-left**



Px and em:

1. **Pixels (px)**:
   * **px** stands for "pixels".
   * It's an absolute unit of measurement.
   * 1 pixel is equal to one dot on a computer screen or display.
   * Pixel values provide precise control over the size of elements.
   * They are commonly used for specifying fixed sizes, such as border widths, padding, and margins.

Ems (em):

* em is a relative unit of measurement.
* It is based on the computed value of the font size of the parent element.
* 1em is equal to the computed font size of the element.
* Using em allows for scalable and flexible layouts, especially when working with responsive designs.
* It's particularly useful when you want sizes to be relative to the font size of the parent element.

Float:

float property in CSS is used to specify whether an element should be floated to the left or right side of its containing element, allowing content to flow around it. It's primarily used for creating layouts where elements are positioned side by side, such as floating images within text or creating multi-column layouts.

Here's a breakdown of how the float property works:

Values: The float property accepts two main values:

* left: The element floats to the left.
* right: The element floats to the right.

Display property:

The **display** property in CSS is used to specify the type of rendering box used for an HTML element. It determines how an element is displayed within the document layout. The **display** property offers various values, each affecting the layout and behavior of the element differently. Here are some common values for the **display** property:

1. **Block level display**: This value causes an element to generate a block-level box, which means it takes up the full width available by default and starts on a new line. Examples of block-level elements include **<div>**, **<p>**, **<h1>** through **<h6>**, and **<ul>**.
2. **Inline display**: This value causes an element to generate an inline-level box, which means it does not start on a new line and only takes up as much width as necessary. Examples of inline elements include **<span>**, **<a>**, **<strong>**, and **<em>**.

Over Flow:

The **overflow** property in CSS is used to control what happens when content overflows its containing element in terms of both visible content and scrollbars. Here are the common values for the **overflow** property:

1. **visible**: This is the default value. Content is not clipped, and it may extend beyond the bounds of the containing element. In other words, overflowing content will be visible outside the container.
2. **hidden**: Content that overflows the container is clipped and hidden. This means any content that extends beyond the boundaries of the container is not visible.
3. **scroll**: If content overflows the container, a scrollbar is provided to scroll the content within the container. Even if the content does not exceed the container's dimensions, scrollbars may still be visible.
4. **auto**: Similar to scroll, but scrollbars are only shown when needed. If the content does not overflow the container, no scrollbars are displayed.
5. **clip**: Content that overflows the container is clipped, meaning it's cut off at the container's boundaries. Unlike **hidden**, where the content simply isn't shown, **clip** removes the overflowed content entirely.

Position:

Positions:

In CSS, the **position** property is used to specify the positioning method of an element within its containing element. There are five main values for the **position** property:

**1.Static**:

* + This is the default value.
  + Elements with **position: static;** are positioned according to the normal flow of the document.
  + They are not affected by the **top**, **right**, **bottom**, and **left** properties.
  + Changing the **position** to **static** is usually not necessary unless you want to override a previously set position.

Relative:

**2.Relative:**

* Elements with position: relative; are positioned relative to their normal position in the document flow.
* When you use top, right, bottom, or left properties with position: relative; it will move the element from its normal position without affecting the layout of other elements.
* If you don't use the offset properties (top, right, bottom, or left), the element remains in its original position.

**3.Absolute**:

* + Elements with **position: absolute;** are positioned relative to the nearest positioned ancestor (an ancestor element with a **position** value other than **static**).
  + If there is no positioned ancestor, the element is positioned relative to the initial containing block (usually the viewport).
  + Absolute positioned elements are removed from the normal document flow, and other elements act as if the absolutely positioned element doesn't exist.
  + Using **top**, **right**, **bottom**, or **left** properties with **position: absolute;** will position the element relative to its containing block.

**4.Fixed**:

* + Elements with **position: fixed;** are positioned relative to the viewport (the browser window), even when the page is scrolled.
  + They do not move when the page is scrolled.
  + Like absolute positioning, fixed positioning removes the element from the normal document flow.

**5.Sticky**:

* + Elements with **position: sticky;** are positioned based on the user's scroll position.
  + They behave like **relative** positioning until the element reaches a specified point, then it "sticks" to that position.
  + The sticky effect is achieved by setting both **top**, **right**, **bottom**, or **left** properties and a **z-index**.

Combinators:

In CSS, combinators are used to define relationships between different HTML elements, allowing you to target elements based on their relationship to other elements. There are four main types of combinators:

1. **Descendant Selector (Whitespace)**:
   1. The descendant selector, denoted by a whitespace character, selects an element that is a descendant of another element.
   2. It matches an element that is inside another element, regardless of how deeply nested it is.
   3. For example, **div p** selects all **<p>** elements that are descendants of **<div>** elements.
2. **Child Selector (>)**:
   * The child selector, denoted by the **>** character, selects an element that is a direct child of another element.
   * It matches an element that is a direct child of the specified parent element.
   * For example, **div > p** selects all **<p>** elements that are immediate children of **<div>** elements.
3. **Adjacent sibling Selector (+)**:

* The adjacent sibling selector, denoted by the **+** character, selects an element that is immediately preceded by a specific sibling element.
* It matches an element that is directly adjacent to another element and shares the same parent.
* For example, **h2 + p** selects the **<p>** element that directly follows an **<h2>** element,

**4.General Sibling Selector (~)**:

* The general sibling selector, denoted by the **~** character, selects all sibling elements that follow a specified element.
* It matches elements that share the same parent and appear after the specified element.
* For example, **h2 ~ p** selects all **<p>** elements that follow **<h2>** elements

Pseudo classes and elements:

In CSS, pseudo-classes and pseudo-elements are selectors that allow you to style elements based on their state or position within the document, even if they don't have specific classes or attributes.

**1.Pseudo-classes**:

* + Pseudo-classes are keywords added to selectors that specify a special state of the selected element. For example, hover, active, focus, etc.

**2.Pseudo-elements**:

* + Pseudo-elements are keywords added to selectors that allow you to style a specific part of an element. For example, before, after, first**-line**, etc.

Background image:

**Background Image (background-image)**:

* This property sets one or more background images for an element.
* You can specify multiple images separated by commas, and they will be layered on top of each other, with the first image closest to the element and the last one farthest.

**Background Attachment (background-attachment)**:

* This property specifies whether the background image should scroll with the content (**scroll**), remain fixed in place (**fixed**), or scroll within the element's padding box (**local**).
* **scroll**: The background image scrolls with the content.
* **fixed**: The background image remains fixed in place, even when the page is scrolled.
* **local**: The background image scrolls with the element's contents, but not the entire page

**Background Size (background-size)**:

* This property specifies the size of the background image.
* You can set the size to **auto** (the default), **cover** (resize the image to cover the entire container), **contain** (resize the image to fit within the container), or specific dimensions (e.g., **50%**, **100px**, **200px 100px**, etc.).

**Background Repeat (background-repeat)**:

* This property specifies how the background image should be repeated if it doesn't fully cover the element's background area.
* Values include **repeat** (repeat both horizontally and vertically), **repeat-x** (repeat horizontally), **repeat-y** (repeat vertically), and **no-repeat** (do not repeat).

**Background Position (background-position)**:

* This property specifies the starting position of the background image.
* You can use keywords like **left**, **right**, **top**, **bottom**, or percentage values (**50% 50%**) to position the image relative to the container.
* You can also use length values (**10px 20px**) to specify an absolute position.