**Grade 8**

**Lesson 1: Introduction to CSS**

**Lesson Objectives**

* Understand what CSS (Cascading Style Sheets) is and how it enhances HTML
* Learn how to apply CSS styles to HTML elements
* Practice styling text, backgrounds, and layout using CSS

**What Is CSS?**

**CSS** (Cascading Style Sheets) is a language used to style and format the layout of HTML elements on a webpage. It allows you to control the look of the content, such as colors, fonts, spacing, positioning, and more.

While HTML provides the structure of a webpage, CSS is used to style it and make it visually appealing.

**CSS Syntax**

A basic CSS rule is made up of:

* **Selector**: The HTML element you want to style (e.g., h1, p, .class, #id)
* **Property**: The style you want to apply (e.g., color, font-size, margin)
* **Value**: The value of the property (e.g., red, 16px, 10px)

selector {

property: value;

}

**Adding CSS to HTML**

There are three main ways to add CSS to an HTML document:

1. **Inline CSS**: Using the style attribute within an HTML tag
2. **Internal CSS**: Placing CSS rules inside a <style> tag in the <head> section
3. **External CSS**: Linking to an external CSS file using the <link> tag

**Example 1: Inline CSS**

Inline CSS is added directly to the HTML tag using the style attribute.

<!DOCTYPE html>

<html>

<head>

<title>Inline CSS Example</title>

</head>

<body>

<h1 style="color: blue;">Welcome to My Website</h1>

<p style="font-size: 18px; color: green;">This is an inline CSS example.</p>

</body>

</html>

**Example 2: Internal CSS**

Internal CSS is placed inside the <style> tag in the <head> section of the HTML document.

<!DOCTYPE html>

<html>

<head>

<title>Internal CSS Example</title>

<style>

h1 {

color: purple;

}

p {

font-size: 20px;

color: darkblue;

}

</style>

</head>

<body>

<h1>Welcome to My Website</h1>

<p>This is an internal CSS example.</p>

</body>

</html>

**Example 3: External CSS**

External CSS is placed in a separate .css file, which is linked to the HTML file.

* **CSS File (styles.css):**

h1 {

color: red;

}

p {

font-size: 18px;

color: black;

}

* **HTML File (index.html):**

<!DOCTYPE html>

<html>

<head>

<title>External CSS Example</title>

<link rel="stylesheet" type="text/css" href="styles.css">

</head>

<body>

<h1>Welcome to My Website</h1>

<p>This is an external CSS example.</p>

</body>

</html>

**CSS Properties**

Here are some common CSS properties used to style text and elements:

* **Color**: Changes the color of text or background

p {

color: blue;

}

* **Font Size**: Adjusts the size of text

p {

font-size: 16px;

}

* **Background Color**: Changes the background color of an element

body {

background-color: lightgray;

}

* **Margin**: Adds space around elements

p {

margin: 20px;

}

* **Padding**: Adds space inside an element (around the content)

p {

padding: 10px;

}

* **Text Align**: Aligns text horizontally (left, center, right)

h1 {

text-align: center;

}

**Exercises**

1. **Style a Heading**  
   Create a heading (<h1>) and style it to be a different color and size.
2. **Style a Paragraph**  
   Add a paragraph (<p>) and change the font size, color, and add some padding.
3. **Create a Page Background**  
   Change the background color of your page using CSS.

**Challenge Task**

Create a webpage with the following styles:

* A heading (<h1>) that is centered and has a red color
* A paragraph (<p>) that has a blue font color, 18px font size, and 10px of padding
* A background color of your choice for the webpage

**Summary**

* **CSS** is used to style HTML elements and enhance the look of a webpage
* **Selectors** are used to target HTML elements, and properties are applied to style them
* CSS can be added in three ways: inline, internal, or external
* Common CSS properties include color, font-size, background-color, margin, padding, and text-align

**Quiz Questions**

1. How do you add CSS to an HTML document?  
   a) With the <style> tag in the <body>  
   b) With the style attribute inside the HTML tag  
   c) With the <link> tag in the <head>  
   d) All of the above
2. Which of the following is a valid CSS property for changing the text color?  
   a) font-size  
   b) color  
   c) background-color  
   d) border
3. How do you link an external CSS file to an HTML document?  
   a) <style src="styles.css">  
   b) <link rel="stylesheet" href="styles.css">  
   c) <css src="styles.css">  
   d) <link href="styles.css" rel="style">
4. What does the padding property do in CSS?  
   a) Adds space around an element  
   b) Changes the font size  
   c) Changes the background color  
   d) Adds space inside an element
5. Which of the following is used to center text in CSS?  
   a) text-align: center;  
   b) font-align: center;  
   c) center-text: true;  
   d) align-text: center;

**Answer Key:**  
1 - d, 2 - b, 3 - b, 4 - d, 5 - a

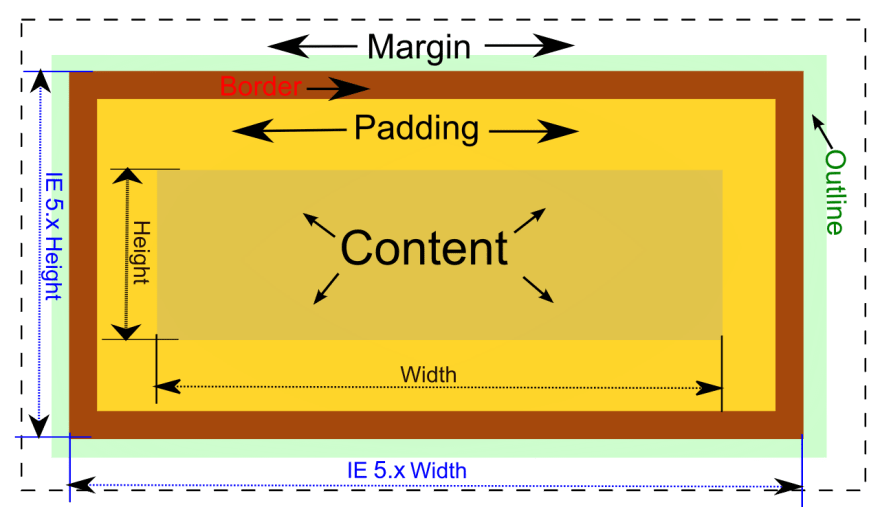
**Lesson 2: CSS Box Model**

**Lesson Objectives**

* Understand the CSS box model and how every HTML element is treated as a box
* Learn the four parts of the box model: content, padding, border, and margin
* Practice using CSS to control spacing and layout

**What Is the CSS Box Model?**

In CSS, every element on a webpage is treated as a **box**. The **box model** describes the space that surrounds each HTML element and helps you control layout and spacing.

The CSS Box Model includes:

**Box Model Parts**

| **Part** | **Description** |
| --- | --- |
| **Content** | The actual text or image inside the element |
| **Padding** | Space between the content and the border |
| **Border** | The line that wraps the padding and content |
| **Margin** | The space outside the border, separating elements |

**Example**

<!DOCTYPE html>

<html>

<head>

<title>Box Model Example</title>

<style>

.box {

background-color: lightblue;

padding: 20px;

border: 5px solid navy;

margin: 30px;

}

</style>

</head>

<body>

<h1>CSS Box Model</h1>

<div class="box">

This is a box with padding, border, and margin.

</div>

</body>

</html>

In this example:

* The content is the text inside the box
* The padding creates space inside the box
* The border is a thick blue line
* The margin creates space around the box

**How the Box Model Affects Layout**

The total width of a box is:

**Total Width = content + padding + border + margin**  
Same for height.

If your content is 200px wide, padding is 10px, border is 5px, and margin is 20px, then:

**Total Width = 200 + 10*2 + 5*2 + 20\*2 = 270px**

**Exercises**

1. **Create a Box**  
   Use a <div> with class box that includes:

* A background color
* 15px padding
* 3px solid border
* 25px margin

1. **Add Multiple Boxes**  
   Make two boxes with different background colors and spacing. Observe how the margin separates them.
2. **Adjust Padding and Margin**  
   Experiment by changing the padding and margin values and see how the box moves and resizes.

**Challenge Task**

Create three boxes on a page:

* First box: blue background, 10px padding, 5px margin
* Second box: green background, 20px padding, 15px margin
* Third box: orange background, 30px padding, 25px margin  
  Style each with a border and make sure they are clearly separated using margins.

**Summary**

* The **box model** is the foundation of CSS layout
* Every element has: **Content → Padding → Border → Margin**
* Use padding to control space inside the box
* Use margin to create space between boxes
* Borders can be styled with width, color, and type (solid, dashed, etc.)

**Quiz**

1. What part of the box model is closest to the text or content?  
   a) Border  
   b) Margin  
   c) Padding  
   d) Content
2. Which property adds space **inside** the border but **outside** the content?  
   a) Margin  
   b) Padding  
   c) Border  
   d) Width
3. What does the margin property do?  
   a) Adds space inside the element  
   b) Adds space between elements  
   c) Adds a border  
   d) Sets the text color
4. If an element has 10px padding, 2px border, and 5px margin, how much space is added around the content?  
   a) 17px  
   b) 10px  
   c) 5px  
   d) 30px
5. Which CSS rule correctly sets padding to 20px?  
   a) padding: 20;  
   b) padding: "20px";  
   c) padding: 20px;  
   d) padding = 20px;

**Answer Key:**  
1 - d, 2 - b, 3 - b, 4 - a, 5 - c

**Lesson 3: CSS Text Styling**

**Lesson Objectives**

* Learn how to change the look of text using CSS
* Style text color, size, spacing, alignment, decoration, and transformation
* Practice applying text styles to different HTML elements

**Why Style Text with CSS?**

HTML text by default looks plain. With CSS, you can:

* Make text more **readable**
* Add **visual interest**
* Match a **design theme**

**Common CSS Text Properties**

| **Property** | **What It Does** | **Example** |
| --- | --- | --- |
| color | Changes the text color | color: blue; |
| font-size | Changes the size of the text | font-size: 20px; |
| text-align | Aligns text (left, center, right) | text-align: center; |
| font-family | Changes the font style | font-family: Arial; |
| font-weight | Makes text bold | font-weight: bold; |
| text-decoration | Underlines or removes underline | text-decoration: none; |
| text-transform | Makes text uppercase or lowercase | text-transform: uppercase; |
| letter-spacing | Adds space between letters | letter-spacing: 2px; |
| line-height | Sets the spacing between lines of text | line-height: 1.5; |

**Example: Text Styling**

<!DOCTYPE html>

<html>

<head>

<title>Text Styling Example</title>

<style>

h1 {

color: darkred;

font-size: 32px;

text-align: center;

}

p {

font-family: Georgia, serif;

font-size: 18px;

line-height: 1.6;

color: darkslategray;

text-transform: capitalize;

}

.highlight {

font-weight: bold;

text-decoration: underline;

color: orange;

}

</style>

</head>

<body>

<h1>Welcome to My Page</h1>

<p>This is an example of styled text using CSS.</p>

<p class="highlight">This text is highlighted with a class.</p>

</body>

</html>

**Exercises**

1. **Style a Heading and Paragraph**  
   Change the color and alignment of a heading, and the font of a paragraph.
2. **Add a Text Class**  
   Create a CSS class that makes text bold, underlined, and blue. Apply it to a <p> or <h2> tag.
3. **Use Text Transformation**  
   Make a paragraph that shows text in all uppercase letters using text-transform.

**Challenge Task**

Create a short web page that includes:

* A centered title in a fancy font
* A paragraph with custom font size and line height
* A highlighted quote with bold, italic, and colored text

**Summary**

* CSS makes text look better and easier to read
* Use properties like color, font-size, text-align, and font-family to change appearance
* You can create CSS **classes** to apply styles to specific parts of text
* Use text-transform, letter-spacing, and line-height to fine-tune readability

**Quiz**

1. Which property changes the text color?  
   a) font-style  
   b) color  
   c) text-style  
   d) font-color
2. What does text-align: center; do?  
   a) Makes the text bold  
   b) Puts the text on the right side  
   c) Centers the text  
   d) Changes text color to grey
3. How do you make text all uppercase?  
   a) text-transform: upper;  
   b) text-uppercase: true;  
   c) text-transform: uppercase;  
   d) font-case: upper;
4. Which property changes the spacing between lines?  
   a) line-height  
   b) letter-spacing  
   c) font-gap  
   d) margin
5. Which CSS rule removes underline from text?  
   a) text-decoration: none;  
   b) text-underline: false;  
   c) font-style: normal;  
   d) text-no-underline: yes;

**Answer Key:**  
1 - b, 2 - c, 3 - c, 4 - a, 5 - a

**Grade 9**

**Lesson 1: CSS Positioning**

**Lesson Objectives**

* Understand the different CSS positioning schemes
* Learn how to position elements using static, relative, absolute, fixed, and sticky
* Apply positioning to create dynamic and responsive layouts

**What Is CSS Positioning?**

CSS positioning allows you to control the layout and placement of elements on a web page. By changing the position property, you can move elements relative to their normal position, their parent, or the viewport.

**Types of Positioning**

1. **Static (Default)**
   * Elements are positioned according to the normal flow of the document.
   * This is the default value; elements are not affected by top, bottom, left, or right properties.
2. **Relative**
   * The element is positioned relative to its normal position.
   * You can use top, right, bottom, and left to move the element.
3. **Absolute**
   * The element is positioned relative to the nearest positioned ancestor (not static).
   * If there is no such ancestor, it uses the document body.
4. **Fixed**
   * The element is positioned relative to the browser window.
   * It stays in place even when the page is scrolled.
5. **Sticky**
   * The element toggles between relative and fixed, depending on the scroll position.
   * It sticks to a given position as you scroll past it.

**Example: Positioning Elements**

<!DOCTYPE html>

<html>

<head>

<title>CSS Positioning Example</title>

<style>

.relative-box {

position: relative;

top: 20px;

left: 30px;

background-color: lightgreen;

width: 150px;

height: 100px;

}

.absolute-box {

position: absolute;

top: 50px;

left: 50px;

background-color: lightcoral;

width: 150px;

height: 100px;

}

.fixed-box {

position: fixed;

bottom: 10px;

right: 10px;

background-color: lightblue;

width: 150px;

height: 100px;

}

.sticky-box {

position: sticky;

top: 0;

background-color: lightyellow;

padding: 10px;

}

</style>

</head>

<body>

<div class="sticky-box">Sticky Header</div>

<div class="relative-box">Relative Box</div>

<div class="absolute-box">Absolute Box</div>

<div class="fixed-box">Fixed Box</div>

<p style="margin-top: 200px;">Scroll down to see the fixed box stay in place.</p>

</body>

</html>

**Exercises**

1. **Create a relative element**
   * Design a <div> with position: relative; and move it 20px down and 30px to the right.
2. **Absolute positioning**
   * Place a child <div> inside a parent <div>. Set the parent to position: relative; and the child to position: absolute; with specific top and left values.
3. **Fixed navigation bar**
   * Create a navigation bar that stays at the top of the page using position: fixed;
4. **Sticky sidebar**
   * Design a sidebar that sticks to the top of the viewport when scrolling using position: sticky;

**Challenge Task**

Build a webpage layout with the following:

* A fixed header that remains at the top
* A sticky navigation menu that sticks below the header when scrolling
* An absolute positioned image inside a relative container
* A fixed footer that stays at the bottom of the viewport

**Summary**

* CSS positioning allows precise control over element placement.
* Different positioning schemes (static, relative, absolute, fixed, sticky) serve various layout needs.
* Combining positioning with other CSS properties enables complex and responsive designs.

**Quiz Questions**

1. **Which positioning value is the default for HTML elements?**

a) relative

b) absolute

c) static

d) fixed

**What does position: absolute; do?**

a) Positions the element relative to its normal position

b) Positions the element relative to the nearest positioned ancestor

c) Positions the element relative to the viewport

d) Keeps the element in place during scrolling

**Which positioning value allows an element to stay in place even when the page is scrolled?**

a) relative

b) absolute

c) fixed

d) sticky

**What is required for position: sticky; to work properly?**

a) The element must have a defined top, left, right, or bottom value

b) The element must be inside a fixed container

c) The element must have z-index set

d) The element must be absolutely positioned

**If a parent element has position: relative; and a child has position: absolute;, where is the child positioned?**

a) Relative to the viewport

b) Relative to the parent element

c) Relative to the document body

d) Relative to its normal position

**Answer Key:** 1 - c, 2 - b, 3 - c, 4 - a, 5 - b

**Lesson 2: CSS z-index**

**Lesson Objectives**

* Understand the purpose of the z-index property in CSS
* Learn how to control the stacking order of overlapping elements
* Apply z-index in practical scenarios to manage element layering

**What Is z-index?**

In CSS, the z-index property determines the stack order of elements along the z-axis (which extends perpendicular to the screen). When elements overlap, z-index specifies which one appears on top.

**Key Points**

* z-index only works on elements with a position value other than static (i.e., relative, absolute, fixed, or sticky)
* Elements with higher z-index values appear in front of those with lower values.
* z-index can take positive, negative, or zero values.
* The default z-index is auto, which follows the order of elements in the HTML.

**Syntax**

element {

position: relative;

z-index: 1;

}

**Example: Overlapping Boxes**

<!DOCTYPE html>

<html>

<head>

<title>z-index Example</title>

<style>

.box {

position: absolute;

width: 150px;

height: 150px;

opacity: 0.8;

}

.box1 {

background-color: red;

top: 50px;

left: 50px;

z-index: 1;

}

.box2 {

background-color: blue;

top: 100px;

left: 100px;

z-index: 2;

}

</style>

</head>

<body>

<div class="box box1">Box 1</div>

<div class="box box2">Box 2</div>

</body>

</html>

**Negative z-index**

You can assign negative values to z-index to place elements behind others.

.element {

position: relative;

z-index: -1;

}

**Stacking Contexts**

A stacking context is a three-dimensional conceptualization of HTML elements along the z-axis. Each stacking context is self-contained; elements within it are stacked according to their z-index values, but this stacking doesn't affect elements outside of the context.

A new stacking context is created when an element is positioned (relative, absolute, fixed, or sticky) and has a z-index value other than auto.

**Exercises**

1. **Create Overlapping Elements**
   * Design three <div> elements with different background colors.
   * Position them to overlap and assign different z-index values to control which one appears on top.
2. **Experiment with Negative z-index**
   * Create a background image and a content box.
   * Use a negative z-index to place the background image behind the content.
3. **Understand Stacking Contexts**
   * Create a parent <div> with a z-index of 1.
   * Inside it, place two child <div> elements with different z-index values.
   * Observe how the stacking order is affected within the parent context.

**Challenge Task**

Design a modal popup window that appears on top of a semi-transparent overlay. Ensure that:

* The overlay covers the entire page content.
* The modal window appears centered and above the overlay.
* Use appropriate z-index values to achieve the desired layering.

**Summary**

* The z-index property controls the vertical stacking order of elements.
* It only applies to positioned elements.
* Higher z-index values bring elements to the front.
* Understanding stacking contexts is crucial for managing complex layering.

**Quiz Questions**

**Which CSS property controls the stacking order of elements?**

a) position

b) z-index

c) display

d) float

**What is the default value of z-index?**

a) 0

b) auto

c) 1

d) -1

**Can z-index be applied to elements with position: static?**

a) Yes

b) No

**If two overlapping elements have the same z-index, which one appears on top?**

a) The one declared first in HTML

b) The one declared last in HTML

c) The one with higher opacity

d) The one with larger dimensions

**What happens when you assign a negative z-index to an element?**

a) It moves in front of other elements

b) It moves behind other elements

c) It becomes invisible

d) It causes an error

**Answer Key:** 1 - b, 2 - b, 3 - b, 4 - b, 5 - b

**Lesson 3: CSS Inline-Block**

**Lesson Objectives**

* Understand the display: inline-block property in CSS
* Learn how inline-block differs from inline and block display values
* Apply inline-block to create horizontal layouts without floats

**What Is display: inline-block?**

The display: inline-block property allows an element to behave like an inline element while retaining the ability to set width and height, similar to a block element. This means elements can sit next to each other horizontally and still have defined dimensions.

**Comparison with Other Display Values**

* **display: inline**:
  + Elements do not start on a new line.
  + Width and height properties have no effect.
  + Top and bottom margins and paddings are not respected.
* **display: block**:
  + Elements start on a new line and take up the full width available.
  + Width and height properties are respected.
* **display: inline-block**:
  + Elements do not start on a new line.
  + Width and height properties are respected.
  + Top and bottom margins and paddings are respected.

This makes inline-block a versatile option for creating layouts where elements need to be aligned horizontally with specific dimensions.

**Syntax**

element {

display: inline-block;

}

**Example: Horizontal Navigation Menu**

<!DOCTYPE html>

<html>

<head>

<style>

.nav {

background-color: #f2f2f2;

padding: 10px;

text-align: center;

}

.nav li {

display: inline-block;

margin: 0 15px;

padding: 10px 20px;

background-color: #ddd;

list-style-type: none;

}

</style>

</head>

<body>

<ul class="nav">

<li>Home</li>

<li>About</li>

<li>Services</li>

<li>Contact</li>

</ul>

</body>

</html>

**Common Use Cases**

* Creating horizontal navigation menus
* Aligning images and text side by side
* Building grid layouts without using floats
* Designing responsive layouts that wrap as needed

**Exercises**

1. **Create a Horizontal List**
   * Design an unordered list where each list item is displayed horizontally using inline-block.
2. **Image and Caption Alignment**
   * Place an image and its caption side by side using inline-block, ensuring both have defined widths and heights.
3. **Responsive Card Layout**
   * Build a set of content cards that align horizontally and wrap to the next line on smaller screens, using inline-block.

**Challenge Task**

Design a responsive photo gallery where each image is displayed as a thumbnail. Use inline-block to arrange the thumbnails in rows that wrap based on the screen size. Add hover effects to enlarge the images slightly when hovered over.

**Summary**

* display: inline-block combines the flow of inline elements with the box properties of block elements.
* It allows for horizontal alignment of elements with defined dimensions.
* Useful for creating layouts without the complexities of floats or flexbox.

**Quiz Questions**

1. **What does display: inline-block allow you to do that display: inline does not?**

a) Set width and height

b) Align elements vertically

c) Float elements

d) None of the above

**Which display value causes elements to start on a new line and take up the full width?**

a) inline

b) inline-block

c) block

d) none

**In which scenario is inline-block most useful?**

a) Creating vertical lists

b) Aligning elements horizontally with specific dimensions

c) Hiding elements

d) Applying animations

**Does inline-block respect top and bottom margins and paddings?**

a) Yes

b) No

**Can inline-block elements wrap to the next line if there's not enough horizontal space?**

a) Yes

b) No

**Answer Key:** 1 - a, 2 - c, 3 - b, 4 - a, 5 - a

**Grade 10**

**Lesson 1: CSS Align**

**Lesson Objectives**

* Understand how to align elements using CSS
* Learn different alignment techniques for text and block elements
* Apply alignment properties to create visually appealing layouts

**Text Alignment**

The text-align property is used to set the horizontal alignment of text within an element.

**Syntax:**

element {

text-align: left | right | center | justify;

}

* **left**: Aligns text to the left.
* **right**: Aligns text to the right.
* **center**: Centers the text.
* **justify**: Stretches the lines so that each line has equal width (like in newspapers and magazines).

**Example:**

<p style="text-align: center;">This text is centered.</p>

**Vertical Alignment**

Vertical alignment can be achieved using various methods, depending on the context.

**1. Using vertical-align for Inline Elements**

The vertical-align property aligns inline or inline-block elements vertically.

**Syntax:**

element {

vertical-align: baseline | top | middle | bottom;

}

**Example:**

<span style="vertical-align: middle;">Aligned Text</span>

**2. Using Flexbox for Vertical Alignment**

Flexbox provides a powerful way to align items both horizontally and vertically.

**Example:**

<div style="display: flex; align-items: center; justify-content: center; height: 200px;">

<p>Vertically and Horizontally Centered</p>

</div>

* align-items: center; centers items vertically.
* justify-content: center; centers items horizontally.

**Centering Block Elements**

To center a block element (like a <div>), you can use margin auto.

**Example:**

<div style="width: 50%; margin: 0 auto;">

<p>This div is centered horizontally.</p>

</div>

**Exercises**

1. **Center Text**
   * Create a paragraph and center its text using text-align.
2. **Vertical Align Inline Elements**
   * Place an image and text side by side, and align them vertically using vertical-align.
3. **Center a Div Using Flexbox**
   * Create a container with a fixed height and center a child element both vertically and horizontally using Flexbox.

**Challenge Task**

Design a card component that has an image at the top, followed by a title and description. Center all the content horizontally, and ensure the card is centered in the viewport both vertically and horizontally using Flexbox.

**Summary**

* text-align is used for horizontal text alignment.
* vertical-align aligns inline elements vertically.
* Flexbox provides a flexible way to align elements both vertically and horizontally.
* Centering block elements can be achieved using margin: 0 auto;.

**Quiz Questions**

**Which property is used to align text to the center?**

a) vertical-align

b) text-align

c) align-items

d) justify-content

**What does margin: 0 auto; do to a block element?**

a) Aligns it to the left

b) Aligns it to the right

c) Centers it horizontally

d) Centers it vertically

**Which Flexbox property centers items vertically?**

a) justify-content

b) align-items

c) text-align

d) vertical-align

**Can vertical-align be used to align block elements vertically?**

a) Yes

b) No

**Which method is best for centering both horizontally and vertically?**

a) Using text-align and vertical-align

b) Using margin: 0 auto;

c) Using Flexbox with align-items and justify-content

d) Using position: absolute;

**Answer Key:** 1 - b, 2 - c, 3 - b, 4 - b, 5 - c

**Lesson 2: CSS Combinators**

**Lesson Objectives**

* Understand what CSS combinators are and how they define relationships between selectors.
* Learn the four primary CSS combinators: descendant, child, adjacent sibling, and general sibling.
* Apply combinators to create precise and efficient CSS selectors.

**What are CSS Combinators?**

CSS combinators are symbols that define relationships between selectors, allowing you to target elements based on their position relative to other elements in the HTML structure.

**Types of CSS Combinators**

**1. Descendant Combinator (space)**

Selects all elements that are descendants (children, grandchildren, etc.) of a specified element.

**Syntax:**

parent descendant {

/\* styles \*/

}

**Example:**

div p {

color: blue;

}

This selects all <p> elements inside <div> elements, regardless of how deeply nested they are.

**2. Child Combinator ( > )**

Selects all elements that are direct children of a specified element.

**Syntax:**

parent > child {

/\* styles \*/

}

**Example:**

ul > li {

list-style-type: square;

}

This selects all <li> elements that are direct children of a <ul> element.

**3. Adjacent Sibling Combinator ( + )**

Selects the element that is immediately after a specified element, and both share the same parent.

**Syntax:**

element1 + element2 {

/\* styles \*/

}

**Example:**

h2 + p {

font-style: italic;

}

This selects the first <p> element that directly follows an <h2> element.

**4. General Sibling Combinator ( ~ )**

Selects all elements that are siblings of a specified element and come after it.

**Syntax:**

element1 ~ element2 {

/\* styles \*/

}

**Example:**

h2 ~ p {

color: gray;

}

This selects all <p> elements that are siblings of an <h2> element and appear after it.

**Exercises**

1. **Descendant Selector Practice**
   * Create a <div> containing multiple nested elements, including <p> tags. Use the descendant combinator to style all <p> elements inside the <div>.
2. **Child Selector Practice**
   * Build a navigation menu using <ul> and <li>. Use the child combinator to style only the direct <li> children of the <ul>.
3. **Adjacent Sibling Selector Practice**
   * Design a section with headings and paragraphs. Use the adjacent sibling combinator to style the first <p> that comes immediately after each <h2>.
4. **General Sibling Selector Practice**
   * Create a series of headings and paragraphs. Use the general sibling combinator to style all <p> elements that follow an <h2>.

**Challenge Task**

Design a blog post layout where:

* Each <h2> heading is followed by multiple <p> paragraphs.
* The first <p> after each <h2> is styled differently (e.g., bold or italic).
* All subsequent <p> elements after the first are styled with a different color.

Use a combination of adjacent and general sibling combinators to achieve this layout.

**Summary**

* CSS combinators define relationships between selectors, allowing for precise targeting of elements.
* The four main combinators are:
  + **Descendant (space)**: Selects all descendants of an element.
  + **Child ( > )**: Selects direct children of an element.
  + **Adjacent Sibling ( + )**: Selects the next sibling element.
  + **General Sibling ( ~ )**: Selects all sibling elements that follow.

**Quiz Questions**

1. **Which combinator selects all elements that are descendants of a specified element?**

a) Child ( > )

b) Descendant (space)

c) Adjacent Sibling ( + )

d) General Sibling ( ~ )

1. **What does the child combinator ( > ) select?**

a) All descendant elements

b) Only direct child elements

c) All sibling elements

d) Elements with the same class

1. **Which combinator would you use to select the first <p> element immediately after an <h2>?**

a) Descendant (space)

b) Child ( > )

c) Adjacent Sibling ( + )

d) General Sibling ( ~ )

1. **What is the difference between the adjacent sibling ( + ) and general sibling ( ~ ) combinators?**

a) No difference; they function the same.

b) Adjacent sibling selects only the next sibling; general sibling selects all following siblings.

c) Adjacent sibling selects all previous siblings; general sibling selects all following siblings.

d) Adjacent sibling selects all siblings; general sibling selects only the next sibling.

1. **Can combinators be combined to create more specific selectors?**

a) Yes

b) No

**Answer Key:** 1 - b, 2 - b, 3 - c, 4 - b, 5 - a

**Lesson 3: CSS Pseudo-classes**

**Lesson Objectives**

* Understand what CSS pseudo-classes are and how they function.
* Learn common pseudo-classes and their applications.
* Apply pseudo-classes to enhance interactivity and styling in web pages.

**What are CSS Pseudo-classes?**

A CSS pseudo-class is a keyword added to a selector that specifies a special state of the selected element(s). Pseudo-classes allow you to apply styles based on user interaction or the element's position in the document tree. citeturn0search0

**Syntax:**

selector:pseudo-class {

property: value;

}

**Common CSS Pseudo-classes**

**1. :hover**

Applies when the user designates an element (with some pointing device), but does not activate it. For example, when a user hovers over a button.

**Example:**

button:hover {

background-color: lightblue;

}

**2. :active**

Applies while an element is being activated by the user. For example, between the times the user presses the mouse button and releases it.

**Example:**

a:active {

color: red;

}

**3. :focus**

Applies when an element has received focus, either from the user selecting it with the mouse or by navigating to it using the keyboard.

**Example:**

input:focus {

border-color: green;

}

**4. :first-child**

Applies to an element that is the first child of its parent.

**Example:**

p:first-child {

font-weight: bold;

}

**5. :nth-child(n)**

Applies to the nth child of a parent element.

**Example:**

li:nth-child(2) {

color: blue;

}

**6. :not(selector)**

Applies to every element that does not match the given selector.

**Example:**

p:not(.intro) {

color: gray;

}

**Exercises**

1. **Hover Effect**
   * Create a button that changes its background color when hovered over using the :hover pseudo-class.
2. **Focus Styling**
   * Design an input field that changes its border color when focused using the :focus pseudo-class.
3. **First Child Styling**
   * Style the first paragraph inside a <div> to have bold text using the :first-child pseudo-class.
4. **Nth Child Styling**
   * In a list of items, change the text color of every second item using the :nth-child(2n) pseudo-class.

**Challenge Task**

Design a navigation menu where:

* Each link changes color when hovered over.
* The active link (the one currently selected) has a different background color.
* The first link has bold text.

Use appropriate pseudo-classes to achieve this styling.

**Summary**

* CSS pseudo-classes allow you to apply styles based on the state of an element.
* Common pseudo-classes include :hover, :active, :focus, :first-child, :nth-child(n), and :not().
* They enhance interactivity and provide more control over element styling without additional classes or JavaScript.

**Quiz**

1. **What does the :hover pseudo-class do?**

a) Styles an element when it is clicked.

b) Styles an element when the mouse pointer is over it.

c) Styles the first child of an element.

d) Styles an element when it gains focus.

1. **Which pseudo-class targets the first child of a parent element?**

a) :first-child

b) :nth-child(1)

c) :first

d) :child(1)

1. **How would you style every third list item in an unordered list?**

a) li:nth-child(3)

b) li:nth-child(3n)

c) li:nth-of-type(3)

d) li:nth(3)

1. **What does the :not() pseudo-class do?**

a) Selects elements that match the selector inside :not().

b) Selects elements that do not match the selector inside :not().

c) Selects all elements.

d) Selects no elements.

1. **Which pseudo-class applies when an input field is selected by the user?**

a) :hover

b) :active

c) :focus

d) :checked

**Answer Key:** 1 - b, 2 - a, 3 - b, 4 - b, 5 - c