Fundamentals of CSS II

Border Properties

1. border-width:

Specifies the width of the border.
Values:

- thin, medium, thick: Predefined keywords for standard widths.
- <length>: Specific length values (e.g., px, em, rem).

Examples:

```
border-width: thin; /* Predefined keyword */
border-width: 2px;  /* Specific length */
```

2. border-style:

Sets the style of the border.
Values:

- none, hidden, dotted, dashed, solid, double, groove, ridge, inset, outset: Different border styles.

Examples:

```
border-style: dashed; /* Dashed border */
border-style: double; /* Double border */
```

3. border-color:

Defines the color of the border.
Values:

- <color>: Specific color values (e.g., named colors, hex codes, RGB values).

Examples:

```
    border-color: #3498db;  /* Hex color code */
    border-color: red;     /* Named color */
```

4. border:

Shorthand property for setting border-width, border-style, and border-color in one declaration.

Example:

```
border: 2px solid #3498db; /* Width, style, and color
                              combined */
```

5. border-radius:

Defines the radius of the corners of an element.
Creates rounded corners when applied.
Values:

- <length>: Specific length values for the radius.
- percentage: Percentage of the box's size.

Example:

```
   border-radius: 10px;      /* Fixed radius */
   border-radius: 50%;       /* Rounded corners, 50% of the box size
                             */
```

6. border-top, border-right, border-bottom, border-left:

Specifies individual borders for each side of an element.
Allows customisation of each border separately.
Values:

- Same as border-width, border-style, and border-color for individual sides.

Example:

```
    border-top: 1px solid #ccc;
    border-right: 2px dotted #555;
```

7. border-image:

Applies an image as the border instead of a solid color.
Useful for creating decorative borders.
Values:

- source: URL of the image.
- slice: Defines how the image is sliced.
- width, outset, repeat: Additional styling properties.

Example:

```
border-image: url('border.png') 30 round; /* Image, slice, and
                                             repeat value */
```

8. border-collapse and border-spacing:

Used in table elements.
border-collapse sets whether table borders should be collapsed into a single border or separated.
border-spacing sets the spacing between adjacent cells' borders.

Values:

- collapse: When border-collapse is set to "collapse," it means that adjacent table cell borders will be combined into a single border. The borders between cells are effectively collapsed, and there is no spacing between them.
- separate: When border-collapse is set to "separate," each table cell maintains its own distinct border. There is spacing between adjacent cell borders, which do not merge into a single border.
- <length>: Sets the spacing between adjacent cells' borders.

Example:

```
border-collapse: collapse;   /* Collapse table borders */
border-spacing: 5px;         /* Spacing between adjacent cell
                             borders */
```

9. box-shadow:

Adds a shadow effect to the entire box, including the border.
Consists of horizontal and vertical offsets, blur radius, spread radius, and color.

Values:

- Horizontal Offset (<length>): Represents the horizontal distance the shadow will be offset to the right. Positive values move the shadow to the right, and negative values move it to the left.
- Vertical Offset (<length>): Represents the vertical distance the shadow will be offset downward. Positive values move the shadow down, and negative values move it up.
- Blur Radius (<length>): Defines the blurring effect applied to the shadow. A higher value results in a more diffuse and softer shadow.
- Spread Radius (<length>): Represents the amount the shadow should expand or contract. Positive values increase the size of the shadow, while negative values decrease it.
- Color (<color>): Specifies the color of the shadow. This can be a named color, a hexadecimal color code, an RGB value, or any valid CSS color representation.

Example:

```
box-shadow: 5px 5px 10px #888888; /* Offset, blur radius,
spread radius, and color */
```

Margin Properties

1. margin:

Sets the margin for all four sides of an element.
Can take one, two, three, or four values to represent the top, right, bottom, and left margins.

Example:

```
margin: 10px;          /* All sides have a 10px margin */
margin: 10px 20px;      /* Top and bottom: 10px, Right and left:
                        20px */
```

```
margin: 10px 20px 15px; /* Top: 10px, Right: 20px, Bottom: 15px,
                           Left: 20px */
```

2. margin-top, margin-right, margin-bottom, margin-left:

Sets the margin for individual sides of an element.
Allows customisation of each side's margin separately.

Example:

```
margin-top: 15px;    /* Top margin */
margin-right: 20px;  /* Right margin */
Negative Values:
```

3. Negative Values

Margins can be set as negative values.
Useful for overlapping elements or fine-tuning layouts.

Example:

```
margin-left: -10px;
Percentage Values:
```

4. Percentage Values:

Margins can be set as a percentage of the containing element's width.
Useful for creating responsive layouts.

Example:

```
margin: 5%;
```

5. Auto Value:

Setting margin to auto horizontally centres a block-level element within its containing element.
Useful for centring elements horizontally.

Example:

```
margin-left: auto;
margin-right: auto;
```

6. Shorthand for Individual Margins:

Shorthand properties like margin-top, margin-right, etc., can be combined for brevity.

Example:

```
margin: 10px 15px 20px 25px; /* Top, Right, Bottom, Left */
```

7. inherit:

Inherits the margin value from its parent element.
Useful for maintaining consistency in spacing throughout the document.

Example:

```
margin: inherit;
```

Padding Properties

1. padding:

Sets the padding for all four sides of an element.
It can take one, two, three, or four values representing the top, right, bottom, and left padding.

Example:

```
padding: 10px;          /* All sides have a 10px padding */
padding: 10px 20px;      /* Top and bottom: 10px, Right and left:
                         20px */
padding: 10px 20px 15px; /* Top: 10px, Right: 20px, Bottom: 15px, Left:
                           20px */
```

2. padding-top, padding-right, padding-bottom, padding-left:

Sets the padding for individual sides of an element.
Allows customisation of each side's padding separately.

Example:

```
padding-top: 15px;    /* Top padding */
padding-right: 20px;  /* Right padding */
```

3. Percentage Values:

Padding can be set as a percentage of the containing element's width.
Useful for creating responsive layouts.

Example:

```
padding: 5%;
```

4. padding: inherit:

Inherits the padding value from its parent element.
Useful for maintaining consistency in spacing throughout the document.
Example:

```
padding: inherit;
```

5. padding: auto:

Not a commonly used value for padding.
In some cases, it can be used to centre a block-level element horizontally within its containing element.
Example:

```
padding: auto;
```

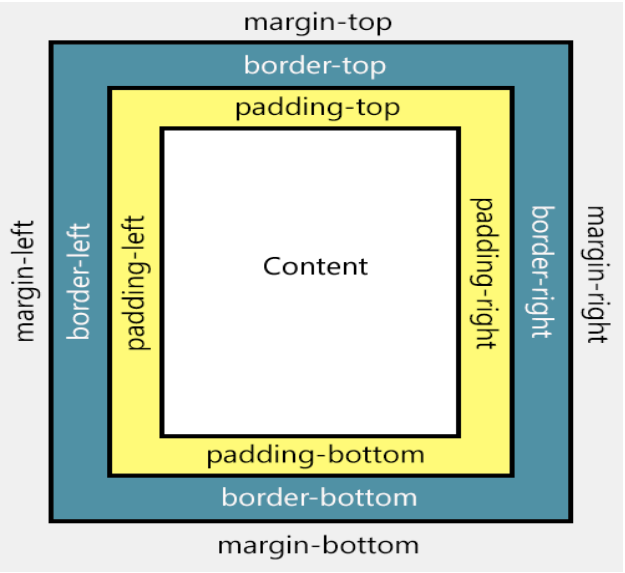
6. Shorthand for Individual Padding:

Shorthand properties like padding-top, padding-right, etc., can be combined for brevity.

Example:

```
padding: 10px 15px 20px 25px; /* Top, Right, Bottom, Left */
```

Margin Vs Padding



Margin:

- Definition:
  Space outside the border of an element, between the element and adjacent elements.
- Affects Box Model:
  Does not affect the size of the element's content area.
- Interaction with Borders:
  Located outside the element's border.
- Purpose:
  Creates space around an element, influencing layout.
- Influence on Layout:
  Indirectly influences the overall layout by defining space between elements.

Padding:

- Definition:
  Space between the content and inner border of an element.
- Affects Box Model:
  Contributes to the total size of the element.
- Interaction with Borders:
  Located inside the element's border.

- Purpose:
  Provides internal spacing for content.
- Influence on Layout:
  Directly affects the size and layout of content within an element.

Box Model

The CSS Box Model is a fundamental concept that defines the layout and structure of elements on a webpage. It comprises four main components:

1. **Content:**

   The actual content of the element, such as text, images, or other media. Defined by the width and height properties.

2. **Padding:**

   The transparent space between the content and the element's border.
   Influences the internal spacing and layout of the content.
   Controlled by the padding property.

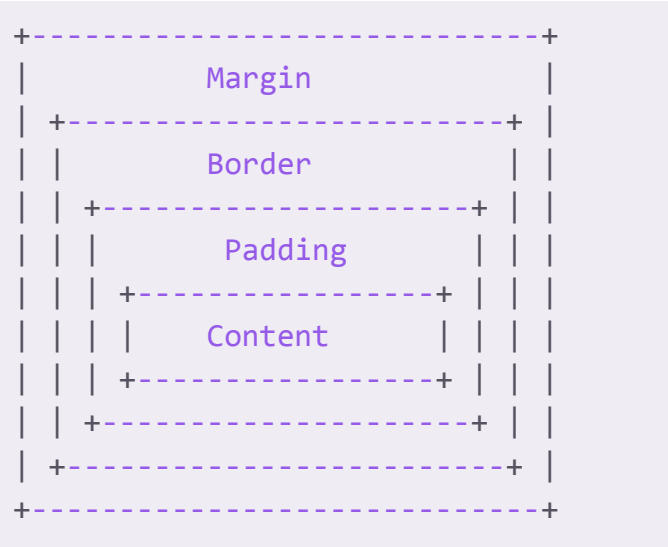
3. **Border:**

   A border surrounding the padding, defining the outer boundary of the element.
   Styled using properties like border-width, border-style, and border-color.

4. **Margin:**

   The transparent space outside the border, creates separation between elements.
   Controls the space between an element and its neighbouring elements.
   Governed by the margin property.

The Box Model can be visualized as follows:



The corrected formula for calculating the total width of the CSS Box Model is:

Total Width = Content width + Border width (left + right) + Padding (left + right)

Box-sizing Property

The box-sizing property in CSS is used to control the sizing behaviour of the CSS **Box Model.**

Default Value: By default, the box-sizing property is set to content-box.

1. **content-box:**
   - Default value.
   - The width and height properties only apply to the content area of the element, excluding padding, border, and margin.
   - The total width and height include only the content.

```
box-sizing: content-box;
```

2. border-box:

- The width and height properties include both the content and the padding but exclude the border and margin.
- It makes it easier to set a specific size for an element without worrying about adding padding and borders to calculate the total width.

```
box-sizing: border-box;
```

Usage:

- Applied to an element, typically in a global style rule.
- Influences how the width and height are calculated.

```
/* Example */
body {
  box-sizing: border-box;
}
```

Benefits of border-box:

- Simplifies layout calculations.
- Easier to create responsive designs, especially when dealing with percentages.
- Intuitive for developers when setting specific dimensions for elements.

Display Properties

The display property in CSS is used to define the layout behaviour of an element.

1. display: inline;

Inline Elements:

- Elements set to display: inline; behave like inline text.
- They flow with the surrounding content and do not start on a new line.

- Width and height properties have no effect, and margins/paddings only affect left and right.

```
span {
  display: inline;
}
```

2. display: block;

Block Elements:

- Elements set to display: block; create a block-level box.
- Starts on a new line and takes up the full width available.
- Width, height, margin, and padding properties are applicable.

```
div {
  display: block;
}
```

3. display: inline-block;

Inline-Block Elements:

- Combines features of both inline and block.
- Flows with the surrounding content like an inline element but allows for setting the width, height, margin, and padding properties like a block element.
- Useful for creating inline elements with block-level styling.

```
img {
  display: inline-block;
}
```

Usage:

Choose the appropriate display value based on the desired layout and styling.

- Inline: Suitable for elements within a line of text.
- Block: Useful for standalone elements that need to start on a new line.
- Inline-Block: Combines inline flow with block-level styling, often used for styling inline elements.

Common HTML Elements:

- and <a> are often styled with display: inline.
- , <p>, and <h1> are commonly styled with display: block;.
- <img> and <button> are examples of elements styled with display: inline-block;.

Some Important Properties:

1. min-width:

Definition:

- Specifies the minimum width an element should have.
- Ensures that the element's width is never less than the specified value.

Usage:

- Commonly used in responsive design to ensure elements do not become too narrow.
- Applied to block-level and inline-block elements.

```
.container {
  min-width: 300px;
}
```

2. max-width:

Definition:

- Specifies the maximum width an element should have.
- Ensures that the element's width does not exceed the specified value.

Usage:

- Useful for preventing elements from becoming too wide, especially in responsive layouts.

- Applied to block-level and inline-block elements.

```
.container {
  max-width: 800px;
}
```

3. min-height:

Definition:

- Specifies the minimum height an element should have.
- Ensures that the element's height is never less than the specified value.

Usage:

- Useful for preventing elements from becoming too short, ensuring a minimum height.
- Applied to block-level and inline-block elements.

```
.box {
  min-height: 100px;
}
```

4. max-height:

Definition:

- Specifies the maximum height an element should have.
- Ensures that the element's height does not exceed the specified value.

Usage:

- Prevents elements from becoming too tall, useful in responsive designs.
- Applied to block-level and inline-block elements.

```
.box {
  max-height: 400px;
}
```

5. vertical-align:

Definition:

- Determines the vertical alignment of an inline or inline-block element to its containing element or another inline element.

Values:

- baseline: Aligns the baseline of the element with the baseline of its parent.
- top: Align the top of the element with the top of the tallest element in the line.
- middle: Aligns the middle of the element with the middle of the line.
- bottom: Aligns the bottom of the element with the bottom of the line.

Usage:

- Commonly used with inline elements within a line of text.
- Useful for aligning elements vertically within a container.

```
.inline-element {
  vertical-align: middle;
}
```

List Styling Properties

1. list-style-type:

Definition:
Specifies the type of marker or style for list items.

Values:

- disc: Default, filled circle.
- circle: Empty circle.
- square: Filled square.
- decimal: Decimal numbers.
- lower-roman: Lowercase Roman numerals.
- upper-roman: Uppercase Roman numerals.
- lower-alpha: Lowercase alphabetical letters.

- upper-alpha: Uppercase alphabetical letters.
- none: No marker.

Example:

```
ul {
  list-style-type: square;
}
```

2. list-style-image:

Definition:
Specifies an image as the marker for list items.
Values:
URL to the image file.
Example:

```
ul {
  list-style-image: url('bullet.png');
}
```

3. list-style-position:

Definition:
Specifies whether the marker should be inside or outside the content flow.
Values:

- outside: Default, the marker is outside the content flow.
- inside: The Marker is inside the content flow.

Example:

```
ul {
  list-style-position: inside;
}
```

4. list-style (Shorthand):

Definition:
Combines list-style-type, list-style-image, and list-style-position in one declaration.

Example:

```
ul {
  list-style: square inside url('bullet.png');
}
```

Table Styling Properties

1. border-collapse:

Definition:
Specifies whether the table borders should be collapsed into a single border or separated.
Values:

- collapse: Borders are collapsed into a single border.
- separate: Borders are separated.

Example:

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

2. border-spacing:

Definition:
Sets the spacing between adjacent cells' borders when border-collapse is set to separate.
Values:
Length values.

Example:

```
table {
  border-spacing: 5px;
}
```

3. table-layout:

Definition:
Specifies the algorithm used to lay out the table cells, columns, and, in some cases, rows.
Values:

- auto: Default, cell size based on content.
- fixed: Cells have a fixed layout, and extra space is divided proportionally.

Example:

```
table {
  table-layout: fixed;
}
```

4. width:

Definition:
Sets the width of the table.
Values:
Length values, percentages, or auto.

Example:

```
table {
  width: 100%;
}
```

5. caption-side:

Definition:
Specifies the placement of the table caption.
Values:
top, bottom, left, right.

Example:

```
caption {
  caption-side: top;
}
```

6. text-align and vertical-align:

Definition:
Controls the horizontal and vertical alignment of table content.
Values:

- left, center, and right for text-align.
- top, middle, and bottom for vertical-align.

Example:

```
th, td {
  text-align: center;
  vertical-align: middle;
}
```

7. background-color:

Definition:
Sets the background color of the table, rows, or cells.
Values:
Color values.

Example:

```
table {
  background-color: #f2f2f2;
}
```

8. border:

Definition:
Sets the border properties for the table, rows, or cells.
Values:
Shorthand for border-width, border-style, and border-color.
Example:

```
table {
  border: 1px solid #ddd;
}
```

Link Styling Properties

1. color Property:

Definition:
Sets the color of the link.

Example:

```
a {
  color: #3498db; /* Blue color */
}
```

2. text-decoration Property:

Definition:
Specifies the decoration applied to the text of the link.
Values:

- none: Default, no decoration.
- underline: Adds an underline.
- overline: Adds a line above the text.
- line-through: Adds a line through the text.

Example:

```
a {
  text-decoration: none; /* No underline */
}
```

3. text-transform Property:

Definition:
Controls the capitalization of text.
Values:

- uppercase: Converts text to uppercase.
- lowercase: Converts text to lowercase.
- capitalize: Capitalizes the first letter of each word.
- none: Default, no transformation.

Example:

```
a {
  text-transform: uppercase; /* Uppercase text */
}
```

4. font-weight Property:

Definition:
Specifies the weight of the font.
Values:
normal, bold, bolder, lighter, or numeric values.

Example:

```
a {
  font-weight: bold; /* Bold text */
}
```

5. cursor Property:

Definition:
Sets the type of cursor to be displayed when hovering over the link.
Values:

- pointer: Displays a pointing hand cursor, indicating a clickable link or button.
- default: Default cursor, usually an arrow.
- crosshair: Displays a crosshair cursor.
- etc.

Example:

```
a {
  cursor: pointer; /* Change cursor to pointer */
}
```

6. outline Property:

Definition:
Specifies the style, color, and width of an element's outline.
Syntax:

```
outline: [outline-color] || [outline-style] || [outline-width] ||
[outline-offset];
```

Values:

- outline-color: Color value (e.g., red, #00ff00, rgba(255, 0, 0, 0.5)).
- outline-style: Style of the outline (e.g., dotted, solid, double).
- outline-width: Width of the outline (e.g., 2px, medium, thin, thick).
- outline-offset: Specifies the space between the outline and the border edge.

Example:

```
/* Example */
input{
  outline: 2px solid #3498db;
}
```

Conclusion

In conclusion, the provided notes on CSS II fundamentals offer a comprehensive overview of crucial styling properties and techniques. The coverage spans border properties, table styling, margin and padding attributes, the box model, display properties, and link styling. Understanding these properties is fundamental for developers to craft visually appealing and responsive web designs. The notes not only delineate the syntax and usage of each property but also emphasize their impact on layout, making them an invaluable resource for creating engaging and user-friendly interfaces. By grasping these fundamentals, developers can wield CSS with precision, ensuring a seamless and aesthetically pleasing user experience across diverse web projects.

In essence, these notes serve as a foundational guide for web developers, enabling them to harness the power of CSS for effective and efficient

styling. The provided insights into properties like box-sizing, min-width, max-height, and outline contribute to a holistic understanding, fostering the creation of web layouts that are not only visually compelling but also structurally sound and user-centric.

References

1. https://developer.mozilla.org/en-US/docs/Web/CSS/border
2. https://developer.mozilla.org/en-US/docs/Web/CSS/margin
3. https://developer.mozilla.org/en-US/docs/Web/CSS/padding
4. https://developer.mozilla.org/en-US/docs/Learn/CSS/Building_blocks/The_box_model
5. https://tailwindcss.com/docs/display#block-and-inline
6. https://developer.mozilla.org/en-US/docs/Web/CSS/list-style
7. https://developer.mozilla.org/en-US/docs/Learn/CSS/Building_blocks/Styling_tables
8. https://developer.mozilla.org/en-US/docs/Learn/CSS/Styling_text/Styling_links