Chapter 21 CSS Tables

21.1 - Managing Borders:

Controlling the appearance of table borders in CSS is straightforward. You can easily define the border style using the BORDER property, specifying attributes like size in pixels, border style (e.g., solid, double, dotted), and a color value as either a name or hexadecimal code. You have the flexibility to assign a distinct border size, style, and color to the table, table headers (th), and table data cells (td). When you apply border styles in this manner, each element will have its own border, resulting in a "double" border appearance. To achieve a single border around the entire table, you can utilize the border-collapse property, which will be explained in the following lesson.

Example:

table, th, td {border: 2px solid black;}

Result:

Sets all tables, th elements, and td elements to have a 2-pixel-wide solid black border.

21.2 - Collapsed Borders:

The BORDER-COLLAPSE property enables you to create a single border around your table while disregarding the outer border properties of table headers (th) and table data cells (td). However, it still allows for interior borders between each header and data cell.

Example:

table.one {border-collapse: collapse;}

table.one, th, td {border: 2px solid blue;}

Result:

Configures the table with class "one" to have a single solid exterior border, 2 pixels wide in blue, and interior borders for th and td elements, also 2 pixels wide in blue.

21.3 - Table Width and Cell Height:

You have the ability to control the overall table width using the WIDTH property and adjust the height of th and td elements using the HEIGHT property. These measurements can be specified in various units such as pixels (px), inches (in), centimeters (cm), millimeters (mm), points (pt), or em. Additionally, the WIDTH property for the table can be expressed as a percentage of the browser window.

Example:

```
table {width: 100%;}

th {height: 50px;}

td {height: 25px;}
```

Result:

Sets the table's width to span the browser window, establishes a height of 50 pixels for table headers, and a height of 25 pixels for table data cells.

21.4 - Table Background Color:

You can independently control the background color of th and td elements by specifying a BACKGROUND-COLOR property for each element. Alternatively, you can set the background color for the entire table using the same property.

Example:

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

th {background-color: green;}

td {background-color: yellow;}
```

Result:

Defines solid 1-pixel black borders for the table, th, and td elements, sets th elements to have a green background, and assigns a yellow background to td elements.

Example:

```
table, th, td {border: 1px solid black; background-color: yellow;}
```

Result:

Sets all borders to a 1-pixel-wide black line and sets the background color for all elements to yellow.

21.5 - Table Text Alignment:

To control the horizontal and vertical alignment of text within your table, you can employ the TEXT-ALIGN property for horizontal alignment and the VERTICAL-ALIGN property for vertical alignment. TEXT-ALIGN options include left, right, and center, while VERTICAL-ALIGN options encompass top, bottom, and center. You can apply the same alignment settings to the entire table by configuring them under the table

element's styling. Alternatively, you can customize th and td elements individually for improved visual presentation or readability.

Example:

table {text-align: left; vertical-align: center;}

Result:

Aligns all elements in the table to the left horizontally and centers them vertically.

Example:

th {text-align: center; vertical-align: top;}

td {text-align: left; vertical-align: bottom;}

Result:

Horizontally centers the text in th elements and positions it at the top of the cell, while horizontally aligning the text in td elements to the left and aligning it with the bottom of the cell.

21.6 - Table Padding:

You can regulate the space between the content and the border of th and td elements by incorporating the PADDING property into your style code. This adjustment enhances the visual clarity and readability of your table.

Example:

th {padding: 10px;}

td {padding: 20px;}

Result:

Establishes a 10-pixel space around the content inside th elements, and a 20-pixel space around the content inside td elements.

Chapter 22 Working with Transforms in CSS

22.1 - Understanding Transforms:

The CSS TRANSFORM property enables you to manipulate your elements by rotating, spinning, scaling, stretching, and moving them in both two and three dimensions. This provides your webpage with enhanced visual appeal and allows precise positioning of elements.

When utilizing the TRANSFORM property, it's important to note that different syntax is required for web browsers like Chrome, Safari, and Opera, as they do not support the standard syntax. To ensure compatibility with these browsers, you must include a prefix before the TRANSFORM property, which is "-webkit-". You can integrate this prefix within the same set of curly brackets as your primary TRANSFORM code. For example, when using the translate method, your code might look like this: {transform: translate(50px, 100px); -webkit-transform: translate(50px, 100px);}.

22.2 - 2D Transforms:

When employing the 2D TRANSFORM property, you have various values at your disposal to make adjustments to your elements. The syntax for performing a 2D transform is as follows: {transform: method(X-axis value, Y-axis value);} (Note: There should be no spaces between the different sections of the code when using transform).

There are eleven distinct methods available for the 2D TRANSFORM property, as listed in the table below.

MethodDescription

translate(x, y) Translates (moves) the element along both the X-axis and Y-axis (x and y defined in pixels).

translateX(n) Translates the element solely along the X-axis (n defined in pixels).

translateY(n) Translates the element solely along the Y-axis (n defined in pixels).

scale(x, y) Scales the element by altering its width and height (x represents the width multiplier, and y represents the height multiplier).

scaleX(n) Changes the element's width only (n describes the width multiplier).

scaleY(n) Modifies the element's height only (n describes the height multiplier).

rotate(angle) Rotates the element to the specified angle (angle is defined in degrees).

skew(x-angle, y-angle) Skews the element along both the X-axis and Y-axis (x-angle and y-angle defined in degrees).

skewX(angle) Skews the element along the X-axis by the defined angle (angle defined in degrees).

skewY(angle) Skews the element along the Y-axis by the defined angle (angle defined in degrees).

matrix(n, n, n, n, n, n) The matrix method combines all 2D transform methods into one by using six parameters containing mathematical functions to enable rotation, scaling, translation, and skewing of elements.

22.2 - 2D Transforms (cont.):

Method:

translate Moves the element relative to its original position.

rotate Rotates the element to the specified angle.

Example:

```
div {transform: translate(50px, 100px); -webkit-transform: translate(50px, 100px);}
```

Result:

Translates (moves) the div element 50 pixels to the right and 100 pixels down from its original position.

Example:

```
div.one {transform: rotate(30deg); -webkit-transform: rotate(30deg);}
```

Result:

Rotates the div.one element 30 degrees clockwise.

22.3 - 3D Transforms:

While 2D transforms interact with the x- and y-axes only, 3D transforms also operate along the z-axis of your webpage. For instance, using the rotate method in 3D at 180 degrees will display the "back-side" or a reverse of your element, as opposed to merely flipping it upside down.

In addition to the primary TRANSFORM property, there are several more properties available for use with 3D transforms, as listed in the following table.

Property	Description
transform-origin	Sets the rule to modify the position of transformed elements.

transform-style Specifies how nested elements are rendered in 3D space.

perspective Specifies the perspective of 3D elements.

perspective-origin Specifies the perspective's bottom position for 3D elements.

backface-visibility Determines whether an element should be visible when it's not facing the screen.

22.3 - 3D Transforms (cont.):

While some methods are shared between 2D and 3D transforms, many differ or require additional information. The table below includes the 3D transform methods.

MethodDescription

translate3d(x, y, z) Defines a 3D translation (x, y, z defined in pixels).

translateX(x) Defines a 3D translation along the x-axis only (x defined in pixels).

translateY(y) Defines a 3D translation along the y-axis only (y defined in pixels).

translateZ(z) Defines a 3D translation along the z-axis only (z defined in pixels).

scale3d(x, y, z) Defines a 3D scale transformation (x, y, z defined by

multipliers of width, height, and depth).

scaleX(x) Defines a 3D scale transformation along the x-axis only.

scaleY(y) Defines a 3D scale transformation along the y-axis only.

scaleZ(z) Defines a 3D scale transformation along the z-axis only.

rotate3d(x, y, z, angle) Defines a 3D rotation (x, y, z, angle defined in degrees).

rotateX(angle) Defines a 3D rotation along the x-axis only.

rotateY(angle) Defines a 3D rotation along the y-axis only.

rotateZ(angle) Defines a 3D rotation along the z-axis only.

perspective(n) Defines a perspective view for a 3D transformed element (n expressed in pixels).

matrix3d(n, n, n, n, n, n, n, n, n, n, n, n, n, n, n, n) Defines a 3D transformation using a 4x4 matrix with 16 values, including mathematical functions.

Method:

rotatex Rotates the element along the x-axis.

Example:

div {transform: rotatex(180deg); -webkit-transform: rotatex(180deg);}

Result:

Displays the div element rotated 180 degrees along the x-axis, making it appear "upside-down."

Chapter 23 Transitions and Animations in CSS

23.1 - Transitions:

Transitions provide you with the capability to alter the state of an element gradually over a specific duration, all without requiring additional tools like Flash or JavaScript. For instance, you can smoothly adjust the width of an element when a user hovers over it. The effectiveness of transitions is closely tied to the timing of the change. By default, the TRANSITION-DURATION property is set to zero, making it crucial to specify a duration; otherwise, your transitions won't take effect. There are multiple transition properties, as detailed in the table below.

You can apply transitions to any property in your HTML code that can be quantified mathematically. For example, you can transition the font-weight of your text but not the font-face. The syntax for transitions is as follows: {transition: property_name duration timing_function delay;}. This example represents a concise way to set all properties in one line (shorthand), where each property is listed separately using the full property name and separated by semicolons within the same set of curly brackets, similar to other CSS coding.

Property Description

transition Specifies the rule for consolidating the settings of the four transition properties into a single property.

transition-property Specifies the rule for naming the property to which the transition is applied (e.g., width, font-weight, height, border, etc.).

transition-duration Specifies the rule for determining the duration of the transition (in seconds (s) or milliseconds (ms)).

transition-timing-function Specifies the rule for defining how the transition's speed is calculated (default is ease; other options include linear, ease-in, ease-out, ease-in-out, cubic-bezier()).

transition-delay Specifies the rule for indicating when the transition will commence (default is 0, in seconds (s) or milliseconds (ms)).

Example (css):

```
div {
  width: 100px;
  height: 100px;
  background: blue;
  transition: width height;
  transition-duration: 2s;
}
div:hover {
  width: 200px;
  height: 200px;
}
```

Result: Sets the div element's dimensions to 100px by 100px with a blue background and configures the transition for width and height changes to occur over a two-second duration when the user hovers over the element.

23.2 - Animations:

Animations empower you to introduce visual appeal to your webpage by creating elements and images that automatically move, spin, change colors, and more. Animations enable you to gradually transform an element from one style to another. You can modify multiple styles as many times as needed within an element. When using animations, you have the flexibility to specify when these style changes will occur, either in percentages (0% to 100%) or using "from" and "to," which are equivalent to the percentage values.

To incorporate animations in CSS, the @keyframes rule must be used. This is where you define the animation by specifying the CSS style properties within the @keyframes rule, facilitating the transition from the current style to the new one.

Similar to transitions, you must define durations for your animations; otherwise, they won't execute. Additionally, it's essential to "bind" an animation to an element, which is achieved by specifying the animation name and duration within the styling rules of your element.

Property	Description
@keyframes	Specifies the rule for defining animations.
animation	Specifies the rule for defining the property for all animation properties except animation-play-state.
animation-name	Specifies the rule for naming the @keyframes animation.
animation-duration	Specifies the rule for determining the duration of one cycle of the animation (in seconds (s) or milliseconds (ms); default is 0).
animation-timing-function	Specifies the rule for describing how the animation progresses over one cycle (default is ease; other options include linear, ease-in, ease-out, ease-in-out, cubic-bezier()).
animation-delay	Specifies the rule for indicating when the animation will commence (in seconds (s) or milliseconds (ms); default is 0).
animation-iteration-count	Specifies the rule for defining the number of times an animation is played (default is 1; you can provide a specific number or use "infinite").
animation-direction	Specifies the rule for determining how the animation should play (normal, reverse, alternate, alternate-reverse; default is normal).
animation-play-state	Specifies the rule for determining whether the animation is running or paused (default setting is running).

23.2 Animations (Cont)

Example (css):

```
div {
  width: 150px;
  height: 150px;
  background: red;
  position: relative;
  animation: myfirst 5s infinite;
}

@keyframes myfirst {
  0% { background: red; left: 0px; top: 0px; }
  25% { background: green; left: 200px; top: 0px; }
  50% { background: #c0c0c0; left: 200px; top: 200px; }
  75% { background: black; left: 0px; top: 200px; }
  100% { background: red; left: 0px; top: 0px; }
}
```

Result:

Defines the div element with dimensions of 150px by 150px, an initial background color of red, and an animation duration of five seconds. The @keyframes rule outlines the animation's sequence: it commences from the original position, then shifts 200px to the right while transitioning to green, proceeds 200px down with a gradual change to silver, shifts 200px to the left while transitioning to black, and finally returns to the starting point with a gradual shift back to red. This entire sequence repeats infinitely.

24.1 - Shorthand Properties

Shorthand Properties in CSS offer a convenient way to consolidate selectors within a single declaration, rather than creating separate declarations for each property. This not only streamlines your coding process but also reduces the overall size of your CSS file. To use Shorthand code, simply separate the selectors with spaces within the declaration. It's crucial to ensure that you provide a value for each selector in the correct order. Failing to do so may lead web browsers to either ignore the declaration or use default settings. Therefore, exercise caution when using Shorthand Properties to achieve your desired results. The syntax for Shorthand Properties is as follows:

{Shorthand Property Name: Selector Selector;}

For instance, you can utilize:

{margin: 2px 1px 3px 4px;}

Instead of:

{margin-top: 2px; margin-right: 1px; margin-bottom: 3px; margin-left: 4px;}

The table below displays the available Shorthand Properties and their respective selectors in the prescribed order:

- Shorthand Property: font
  - Selectors: font-size; line-height; font-weight; font-style; font-family

- Shorthand Property: background
  - Selectors: background-color; background-image; background-repeat; background-position

- Shorthand Property: list-style
  - Selectors: list-style; list-style-type; list-style-position; list-style-image

- Shorthand Property: border
  - Selectors: border-width; border-color; border-style

- Shorthand Property: margin

  - Selectors: margin-top; margin-right; margin-bottom; margin-left

- Shorthand Property: padding

  - Selectors: padding-top; padding-right; padding-bottom; padding-left

It's important to note that margin and padding Shorthand properties behave slightly differently due to the four-sided nature of these properties. This can result in various combinations of values. For example, you can specify different values for each of the four margins (top, right, bottom, and left) as shown in the previous example. Alternatively, you might have only two distinct values for all four sides. This shorthand notation can accept one, two, three, or four values. When specifying four values, they are assigned to the respective sides in a clockwise order (top, right, bottom, left). If you provide only two or three values, the missing side receives the same value as the opposite one. And if you specify a single value, it applies to all four sides. For example, the declaration "margin: 3px 1px" would assign a 3-pixel value to the top and bottom margins and a 1-pixel value to the left and right margins, as only two values are listed in the declaration.