Extra notes - CSS - Column Layouts

- Dr Nick Hayward

A brief introduction to CSS column layout and options.

Contents

Intro A **Cascading Style Sheet**, or CSS, allows us to define stylistic characteristics for our HTML. In effect, it helps us define how our HTML is displayed and rendered. The colours used, font sizes, borders, padding, margins, links, and so on.

Initial concepts of column layouts CSS provides various options for content layout relative to *Grids*, *Flexbox*, &c.

However, we may also consider an alternative layout option based on rendering columnar content.

In effect, CSS columns will break up the content in the defined container element, including any descendant elements, into columns.

For example, we may add `column-count` or `column-width` to an element, thereby defining a multi-column container for its content. Each column is now an anonymous *box*.

Multi column containers We may create an initial column based container using either `column-count` or `column-width` .

column count For example, the `column-count` property may be used to define the number of columns to display in the container element.

As the CSS is processed, the web browser will then calculate and assign the required dimensions per column to create the required layout in the container.

For example, we may define a standard `<div>` element as the required column container with a `<p>` for the content.

```html
<div class="col-container">
    <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labo
</div>
```

We may then define a ruleset for this paragraph using the defined `class` attribute.

```css
.col-container {
    column-count: 2;
}
```

The container content, a `<p>` in this example, is rendered with default properties such as margins. This default rendering will, of course, include a margin at the top of each paragraph, which will cause the first column to be rendered with a margin as well at the top.

This default rendering causes a disparity between the top of the first column and the remaining columns in the container. This is due to the container creating a new *Block Formatting Context*, which causes the margins of child elements to *not* collapse with any defined container margins.

We can, of course, override this default behaviour by assigning a class to the first paragraph, and then set the value of the property `margin-top` to `0` .

```
p.col-p {
    margin-top: 0;
}
```

Another option might include an initial heading for the content, similar to a magazine or newpaper headline.

column width We may also choose to set an optimum, preferred width for each column box. We may use this option in place of the previous explicit `column-count` property.

A value for the property `column-width` will cause the web browser to calculate the necessary number of columns to fit the content into columns of the preferred width.

For example, if we modify our previous ruleset

```
.col-container {
    column-width: 150px;
}
```

The container content is still rendered with the same default margins for paragraphs &c., as seen with `column-count` . So, again, we may add various updates to remove this top margin, if necessary.

column count and width combined We may also combine column count and width, effectively defining a preferred width per column until the column count is reached.

When combined, the `column-count` property acts as a maximum value for columns in the container.

For example,

```
.col-container {
    column-count: 3;
    column-widht: 150px;
}
```

In this ruleset, even if there is enough space for more than `3` columns of `150px` each, the container will restrict the columns to the maximum count of `3` .

Likewise, if there is not enough space for `3` columns in the container it will restrict the number of columns relative to width in the container.

columns shorthand We may also use a `columns` shorthand for both `count` and `width` , and combined values as well.

For example,

```
.col-container {
    columns: 3;
}
```

This is the equivalent of using `column-count` with an implicit property set. Likewise, we may use the same syntax for a shorthand `column-width` .

If we combine both properties, we may define the ruleset and `columns` property as follows,

```
.col-container {
    columns: 3 150px;
}
```

As expected, the first value is `columns` and the second is `width` .

Style columns Each column is calculated and generated by the browser as part of the processing of CSS. This means each generated column is rendered as an anonymous box.

As such, we may not style a box specifically. However, we may style the space rendered between columns.

For example, we might use the `column-gap` property to modify the space between the rendered columns.

```
.col-container {
    column-count: 4;
    column-gap: 50px;
}
```

We may also use percentage values for the column-gap to ensure we do not create overly narrow column widths.

column rules Another option is to use one of the available column rules to help add some semblance of custom styling to generate columns.

We may use the following properties,

- `column-rule-color`
- `column-rule-style`
- `column-rule-width`

or the shorthand combined property `column-rule` . A useful benefit of these properties is that we may values we might apply for CSS `border` properties.

In effect, any values supported by the `border` properties may be used with column rules.

For example,

```
.col-container {
    column-count: 4;
    column-gap: 50px;
    column-rule-width: 2px;
    column-rule-style: solid;
    column-rule-color: #A3CDD9;
}
```

These properties are then applied to the overall container, which means the styles are applied evenly to each generated column. Each style is only rendered between columns, and not the outer left and right edge of the columns.

These rules do not modify the standard box model, which means the columns will be modified for width to accomodate a broade column rule width. If the width is wider than the rendered gap, for example, the rendering will overlap the column's content. In effect, the rule will be shown under the column's content.

Other options There are also other options available for rendering multi column layouts with CSS.

For example, we might add a property to define an element as spanning multi columns. We might also choose to add content breaks in columns, &c.

However, whilst the above examples are, generally, well supported by modern web browsers, such extra options are still in development as part of early specifications.

You may notice unexpected rendering results for such extra options, which need to be considered relative to a final page release.

In effect, use such extra options with caution.

References

- MDN - CSS
- MDN - CSS - Multi Columns
- W3 CSS
- W3 Schools - CSS