Extra notes - CSS - Basic Layout - Block and Inline

- Dr Nick Hayward

A brief introduction to the basics of block and inline elements relative to CSS layout usage and rendering.

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.

CSS is used for defining how documents will be presented to users, for example rendered HTML in a web browser.

Block and inline layout Customarily, we consider *block* and *inline* relative to HTML elements, in particular pre-HTML5 usage contexts.

This consideration, however, is defined relative to the element's rendering on the page. In effect, it is a consideration of the markup for rendering purposes, and not *semantic*. This is a noted difference between element groupings in HTML4 and HTML5.

We may, therefore, consider how *block* and *inline* elements will behave as part of the *normal* rendered flow of a document.

normal flow Defined in CSS specifications, version 2.1, *normal flow* may be used to explain the position and rendering of any boxes as part of a *formatting context*.

In effect, such boxes may be either *block* or *inline*, but not both at the same time.

A *Block-level* box may, therefore, participate in the rendered block formatting context. Likewise, *inline-level* boxes will participate in the document's inline formatting context.

The CSS specification also describes how such elements will be formatted and rendered.

block formatting context In this context, *boxes* are arranged in a vertical order, one after another, beginning at the top of a containing box.

We may then use the CSS style property `margin` to define the vertical distance between sibling boxes, effectively creating a sense of separation in the rendered layout.

However, such vertical margin properties will collapse between adjacent block-level boxes in this context.

inline formatting context By contrast, this context arranges *boxes* in a horizontal order, one after another, beginning at the top of the containing box.

Additional CSS style properties may be defined for horizontal margins, borders, and padding, which are preserved and respected between the context's boxes.

A grouping of such boxes in this context is referenced as a *line box*.

We may also align such boxes on the vertical axis using a variety of supported CSS options and layouts.

Block elements and context In a language such as English, we write by default in a horizontal mode. Lines are then arranged in a vertical layout, one on top of the other down the page.

This standard layout practice creates an *inline direction* from left to right on the horizontal axis, and a *block direction* on the vertical axis.

Our use of such block and inline elements follows this practice by default, and is rendered as such on the page.

If we define two block elements, `<p>` paragraphs in this example, each containing text content, we may see a simple, default block layout for these elements.

For example,

```
<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor
incididunt ut labore et dolore magna aliqua. Vulputate sapien nec sagittis aliquam
malesuada bibendum arcu vitae.</p>

<p>Sem integer vitae justo eget magna fermentum iaculis eu non. Eget est lorem ipsum
dolor sit amet consectetur adipiscing elit.</p>
```

As noted in the CSS specification, the rendered separation between these block elements is defined by the default *margins*. The default CSS cascade will add a margin for the top and bottom of each of these block elements. We may, of course, modify such CSS properties, but the default structure and usage is clearly defined.

If we view this initial rendered content, we may see that the separation of content is, indeed, defined by the use of top and bottom margins. Default padding or border properties are not added for the *box model* of each block-level element.

We may, however, choose to update some initial properties for this block element,

```
p {
    border: 3px solid #A5BB87;
}
```

The new border colour will clearly show this separation of content defined by the default margin property.

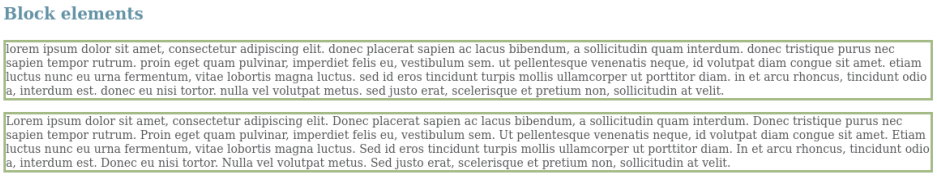

Figure 1: CSS Block Elements - default margins with border

update default usage By default, each block element will use all of the available space in an *inline* direction. In the current example, each `<p>` paragraph element will fill all of the space available on the horizontal axis within its current container.

However, we may also choose to update the default width for the block element.

For example,

```
p {
    width: 300px;
}
```

The layout will continue to adhere to the pattern of stacked boxes on the vertical axis, and will initially align on the left side of the containing box relative to the current writing mode.

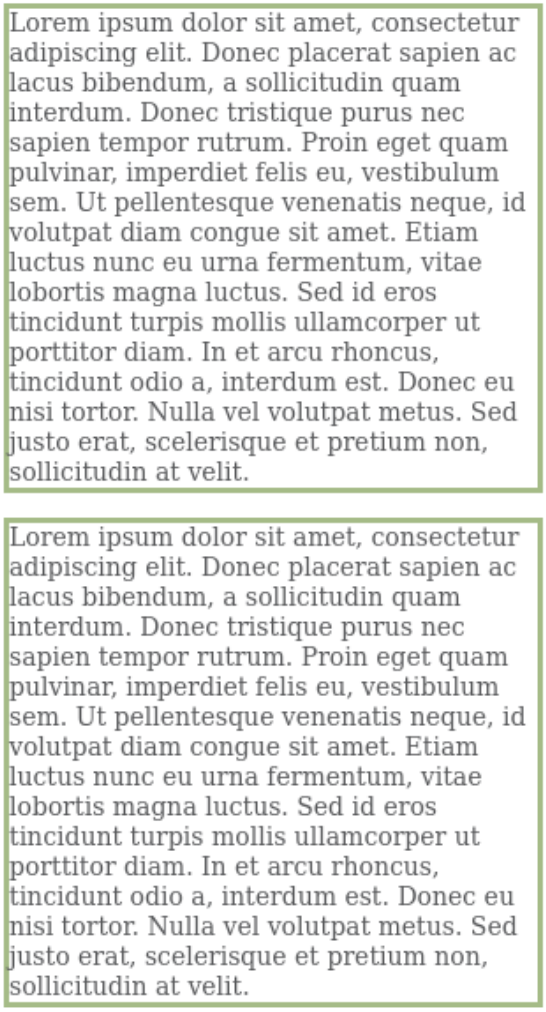

Figure 2: CSS Block Elements - modified width

margins collapse As noted above, and in the CSS specifications, margins for block elements effectively collapse. Whilst this may sound counter-intuitive, considering the previous example, there is a simple rule that CSS follows for such element margins.

For example, if the block element includes a top and bottom margin, and the next element is also block-level with margins, the margin will collapse. In effect, instead of adding both the top and bottom margins of two adjacent block elements, the margins will collapse, and the greater value of the two adjacent margins will be used for margin rendering on the page.

If we consider the following CSS,

```
p {
    margin: 25px 0 50px 0;
}
```

we have now modified the top and bottom margins for our `<p>` paragraph elements to `20px` and `50px` respectively. If we render adjacent paragraphs, these margins will collapse, and the space between the paragraphs will become `50px` , the greater of the two defined margin values.

If we render the updated example, we may clearly see that the space between the two paragraphs is now double the margin above the first paragraph.

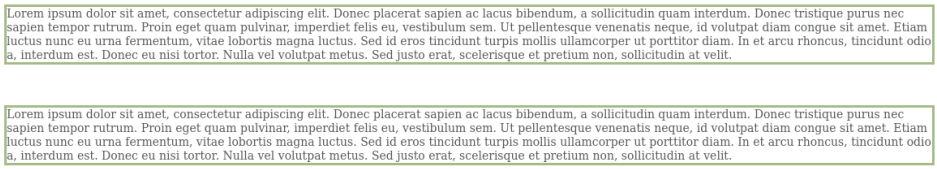

Figure 3: CSS Block Elements - margin collapse

Inline elements and context Inline elements will be rendered in the direction defined for content flow by the current defined writing mode. For example, from left to right for each line for Western languages such as English.

Whilst such elements do not have boxes that stack on the vertical axis, as with block elements, they do have boxes for the element's content.

Each inline box will correspond to the defined inline element, and may be presented one after the other where defined in the markup.

If there are multiple adjacent inline elements, they will be rendered from line to line to fit the available space in the container box.

These created boxes are then known as *line boxes*.

For example, if we define the following HTML markup

```
<p>lorem ipsum dolor sit amet, consectetur adipiscing elit. <span class="inline">donec
placerat sapien ac lacus bibendum</span>, a sollicitudin quam interdum. donec
tristique purus nec sapien tempor rutrum.</p>
```

the rendered content will use a line box for each rendered line. So, for the available space in the containing box this content will be rendered in two line boxes.

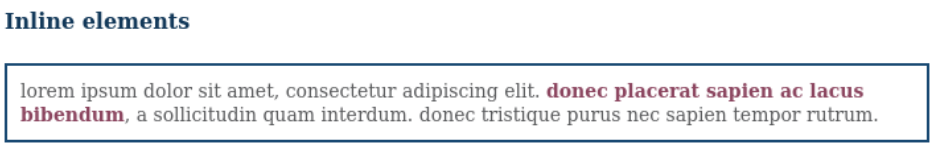

Figure 4: CSS Inline Elements - line boxes

If we modified the width of the page, for example, the number of lines rendered, and the subsequent line boxes will then be updated as well to fit the available space.

The inline boxes, before and after the `<span>` element, are known as *anonymous boxes*. These boxes are added to the rendered content to ensure the lines are rendered in the expected wrapped box. However, we may not target these boxes directly.

These boxes also help define the height of the line boxes in the *block* direction. In effect, the tallest of the boxes in a line will also determine the height of the line box itself.

Override default formatting context In CSS, we may also choose to override the default formatting context for an element.

In effect, we may define an inline element as *block*, and vice-versa.

If we update the last example, overriding the default for the inline element

```
.inline-override {
    display:block;
}
```

we can see how the `<span>` element is now displayed in a block direction box, breaking the flow of the paragraph's content.

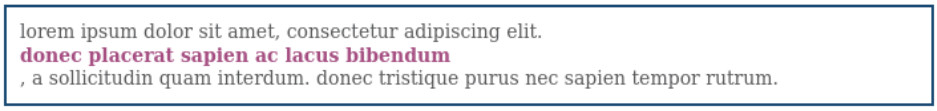

Figure 5: CSS Inline Elements - override defaults

References

- MDN - CSS
- MDN - CSS - Block and inline layout in normal flow
- W3 CSS
- W3 Schools - CSS