Box Model

Every HTML element has a default display value depending on what type of element it is. The default display value for most elements is block or inline.

Block-level Elements

A block-level element always starts on a new line and takes up the full width available (stretches out to the left and right as far as it can).

Examples of block-level elements:

* <div>
* <h1> - <h6>
* <ul>, <ol>
* <p>
* <form>

Inline Elements

An inline element does not start on a new line and only takes up as much width as necessary.

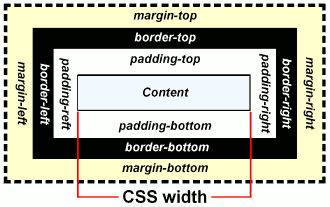
Examples of inline elements:

* <span>
* <a>
* <img>

What is the difference between inline and block elements?

For example, the HTML <div> element is a block level element that can be used as a container for other HTML elements. The <div> element has no special meaning. It has no required attributes, but style and class are common. Because it is a block level element, the browser will display line breaks before and after it. There won’t be any other elements next to it, whether its width occupies fully of what’s available to it or not.

You can add border, margin or padding to both kinds of elements. Since inline elements are part of the document flow, for margin and padding, only left and right sides would have effects for spacing. Top and bottom sides won’t have any effects.



lesson1\_reset\_before.html

Browsers have default styles built-in for HTML elements. For example, the font color is black, and the background color always starts with white. For new elements from HTML5, you can see that they don’t come with padding or margin. For older elements, like <h1>, <p> or <ul>, you can see that they have some padding and margin that come from the browser default styles.

You can also see that <body> comes with some padding (in some browsers, even margin) around it.

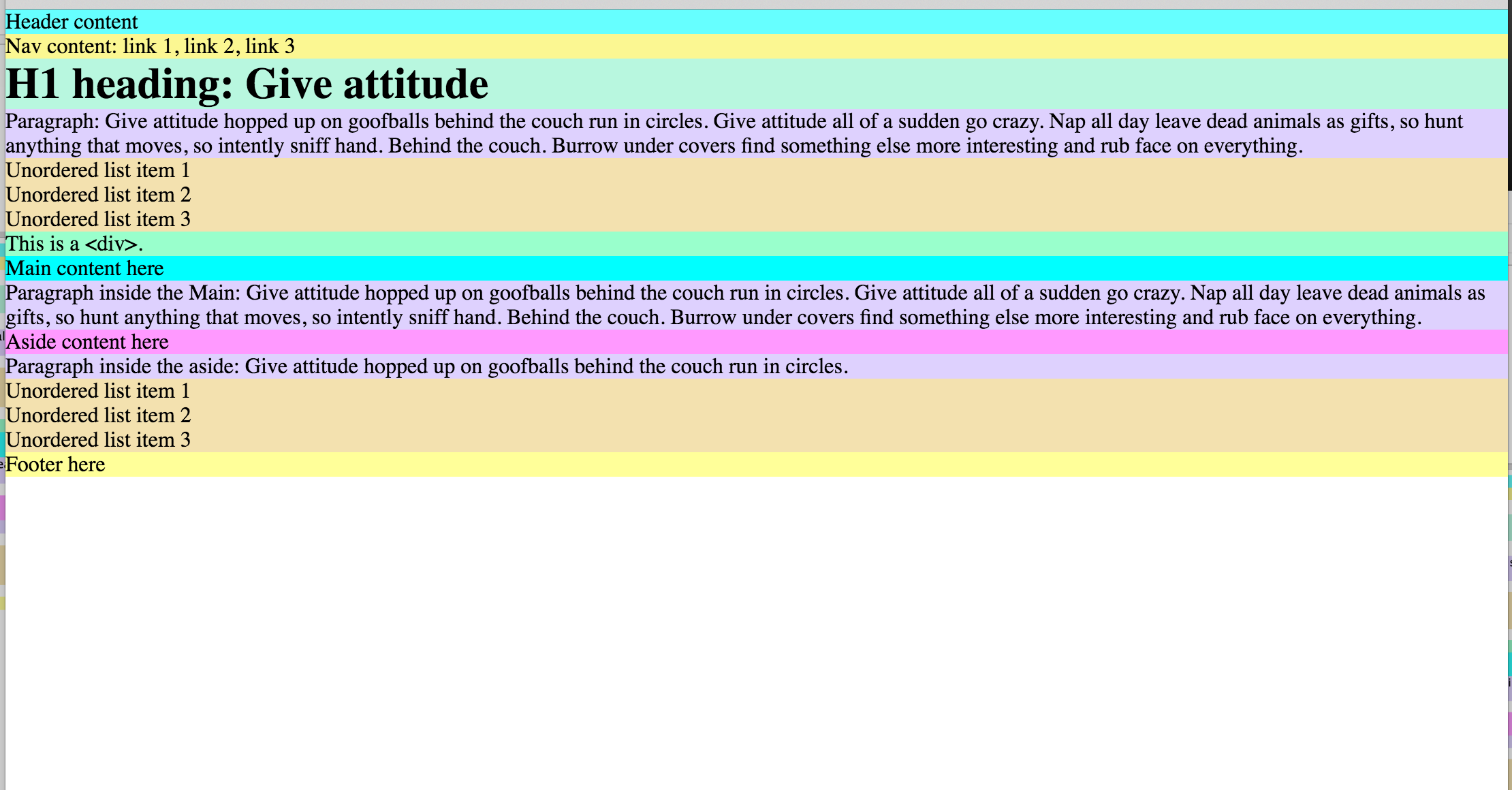
Why is this important? Sometimes when you use <ul> to make navigation menu, there is some spacing issue that you can’t get rid of, that would be the <ul> top margin that is sticking out. If you set it to 0, then you can align things perfectly. Good alignment is part of good web design.



lesson2\_reset\_after.html

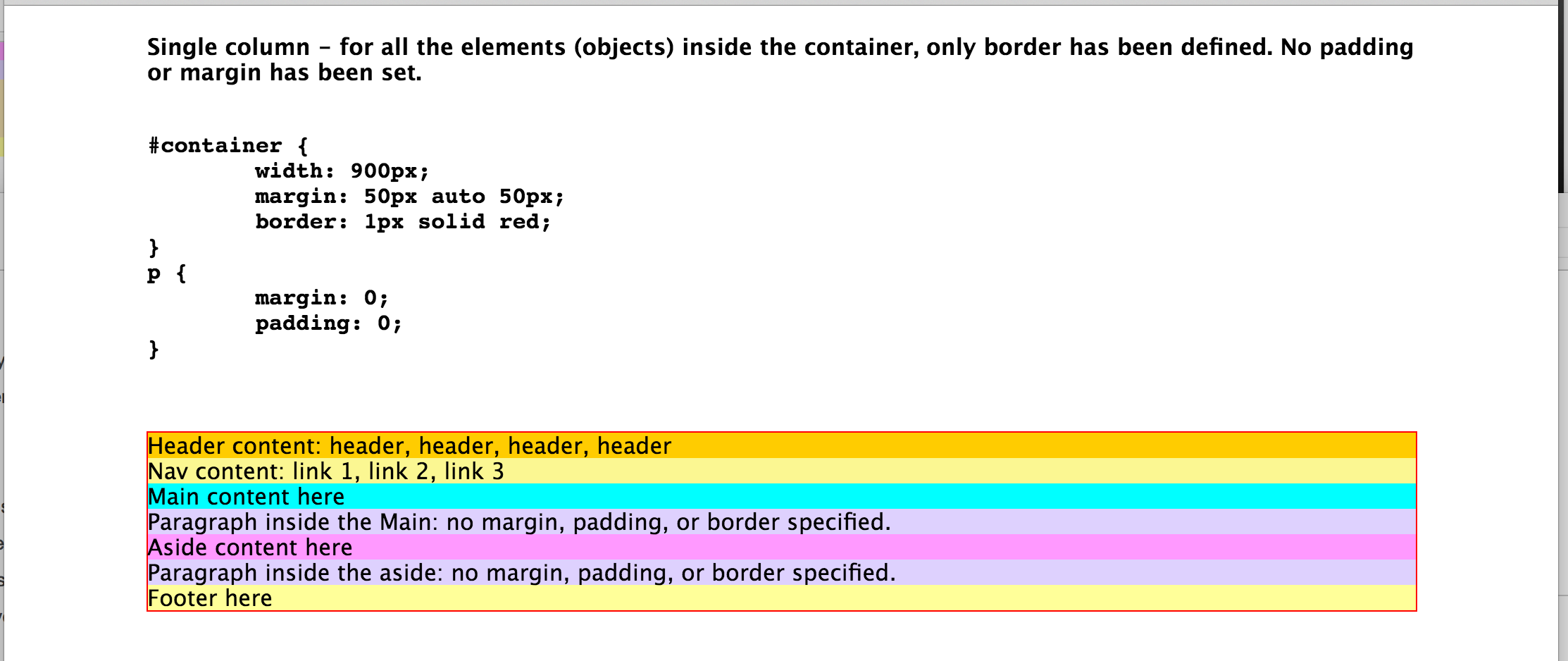
In this page, we set the margin and padding of <body>, <h1>, <p> and <ul> to 0. You can see that things are aligning better. This gives us more controls on the alignment.

For all the subsequent pages, we reset those elements.



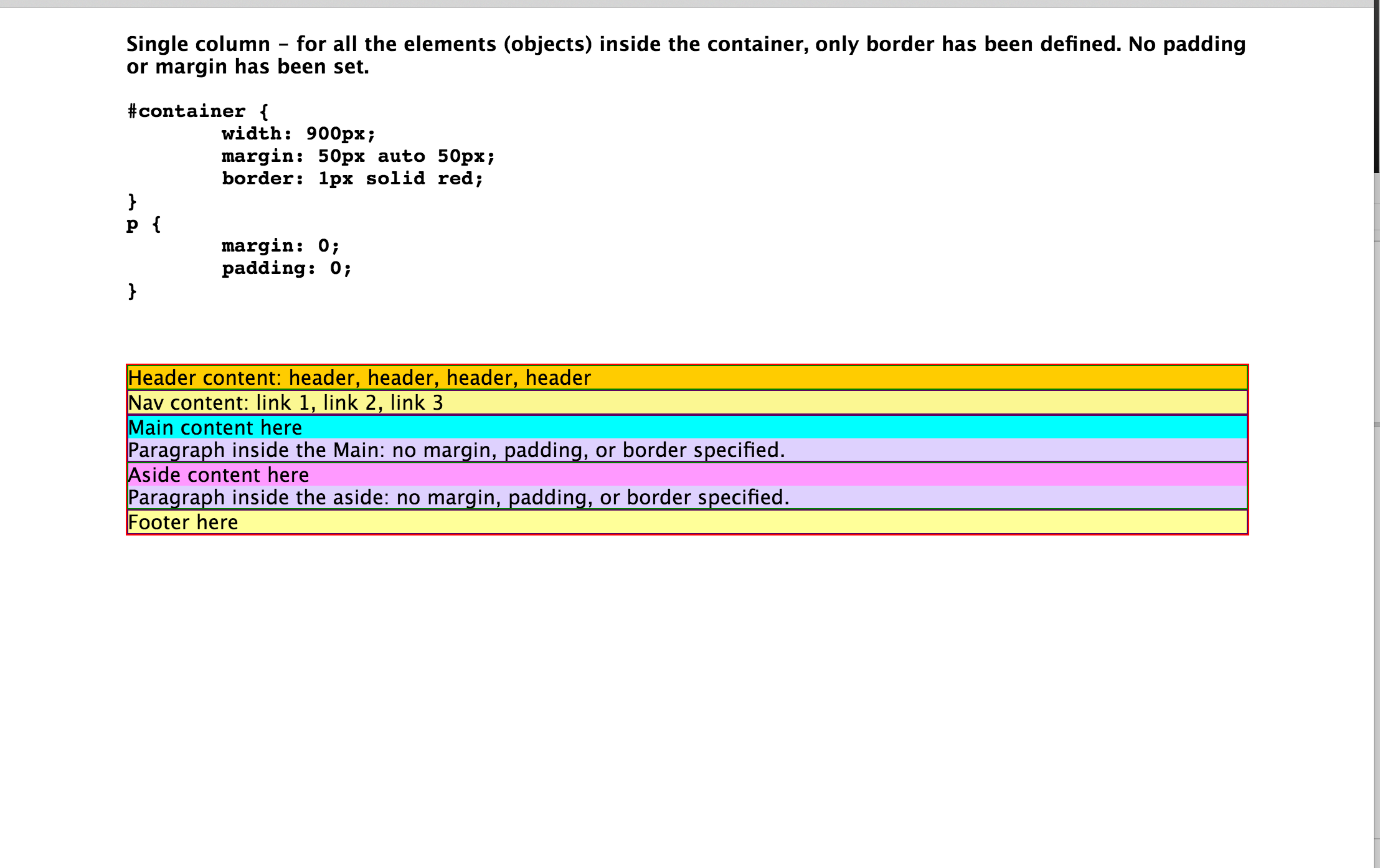
lesson3\_1Col.html

We are starting to build a web page. We have a container (<div id=”container”>) that is 900px wide and perfectly centered. We added a red border so we can see it better. No styles have been added to any elements inside it.



lesson4\_1Col\_border.html

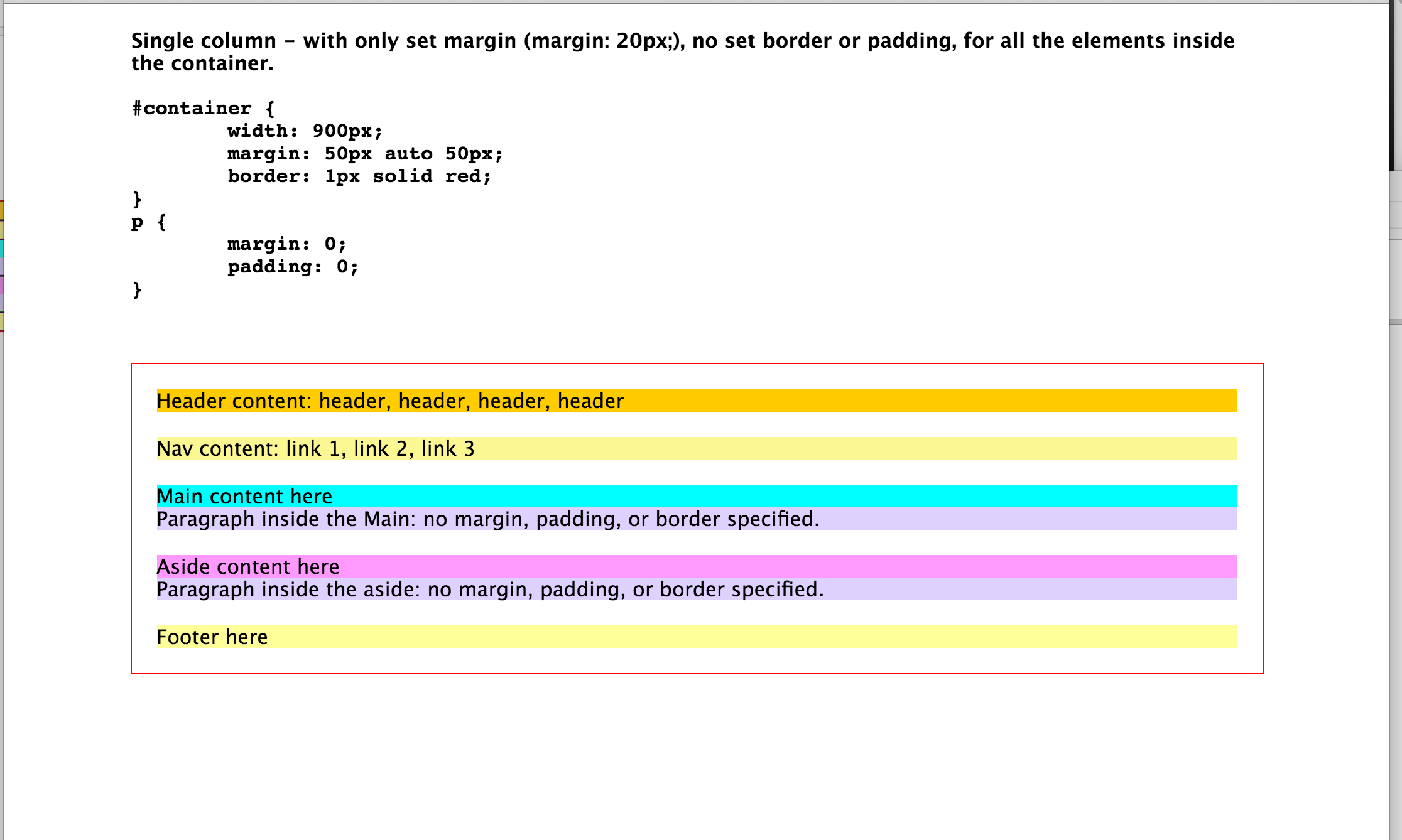
Now we only add borders to all the elements inside the container.



lesson4\_1Col\_margin.html

Now we only add 20px of margin (on all 4 sides) to all the elements inside the container. There is a concept of collapsing margins.

For example, in our current layout, let’s say <header> is changed to have 40px on all sides and <nav> is changed to have 20px on all sides, the distance between <header> and <nav> will be 40px, not 60px. The margin-bottom (40px) of <header> and the margin-top (20px) of the <nav> are overlapping. The longest one wins and dictates how big the space is between them.



lesson4\_1Col\_padding.html

Now we only add padding to all elements inside the container. No margin or border has been added. You can see the spacing from the texts to its edge.



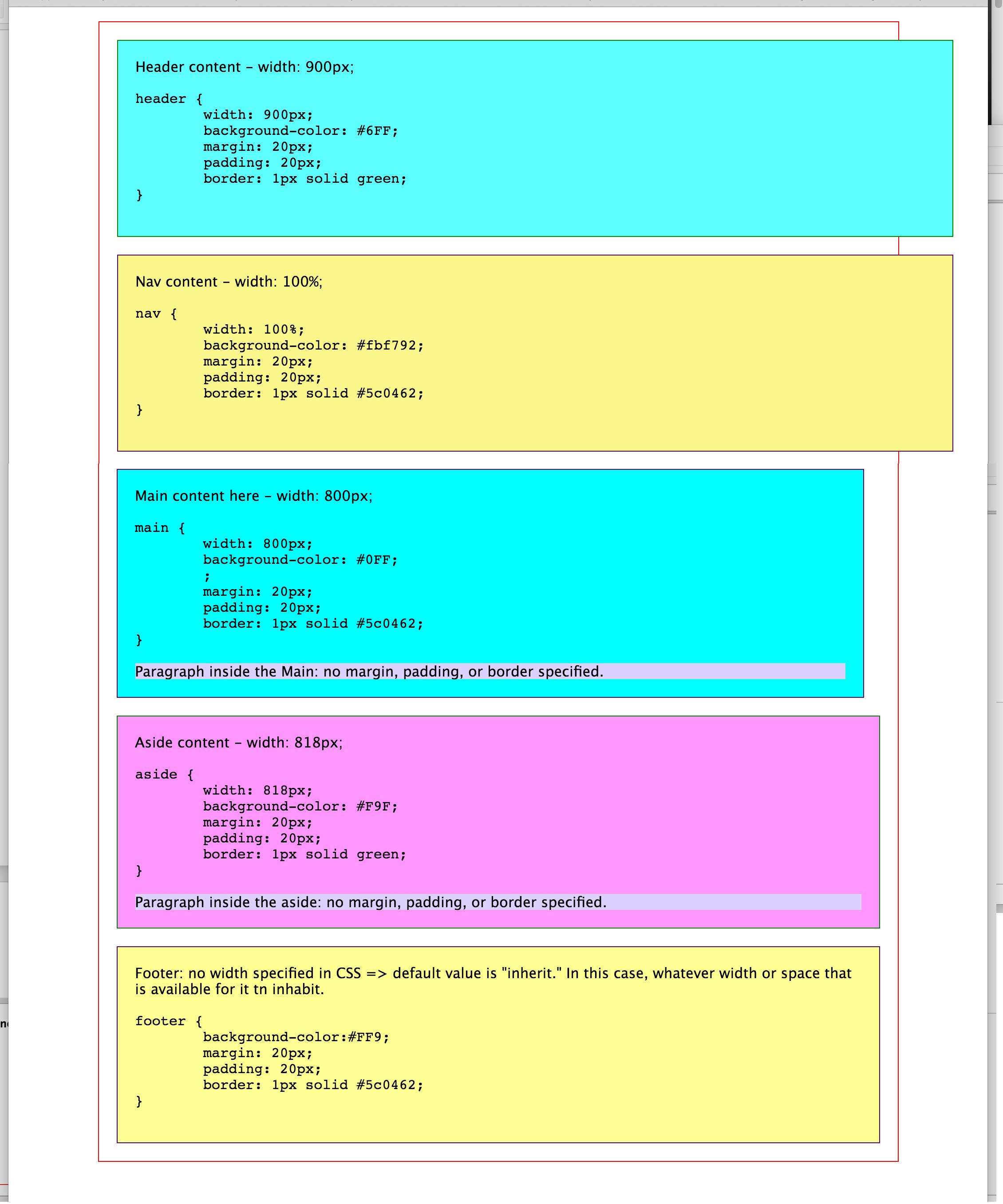
lesson5a\_1Col\_p\_m\_b\_width.html

In this page, we add padding, margin, border, along with widths specified.

You can see some of the elements, after adding those styles, they become too big to fit in the container and are sticking out.

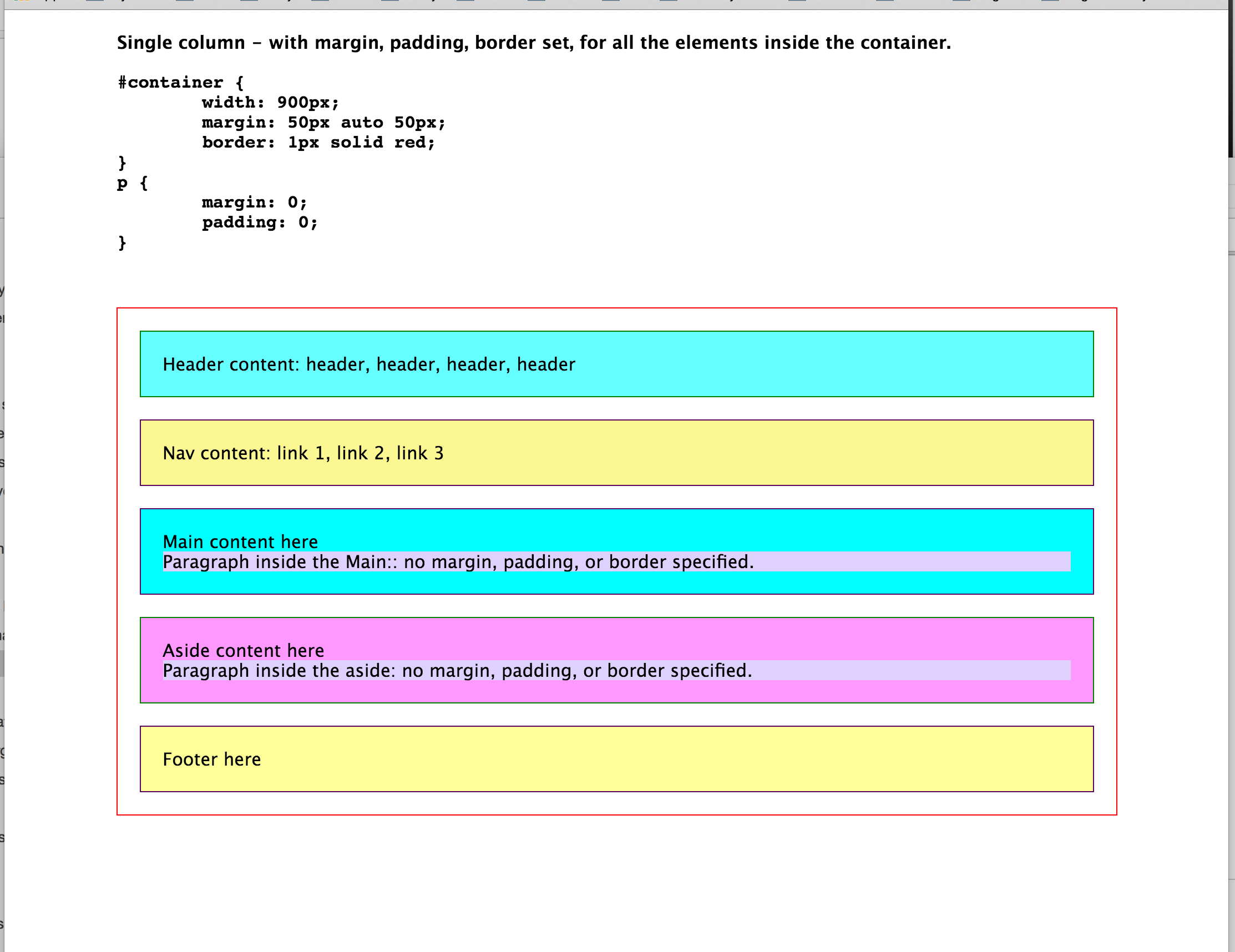
For the width property, width: 100% means 100% of the width (declared width) of whatever container that contains it. In this case, that will be 900px.

When you don’t define a width, then the browser will try to fit the element in perfectly and do the calculation for you, like the <footer>.



lesson5b\_1Col\_p\_m\_b\_noWidth.html

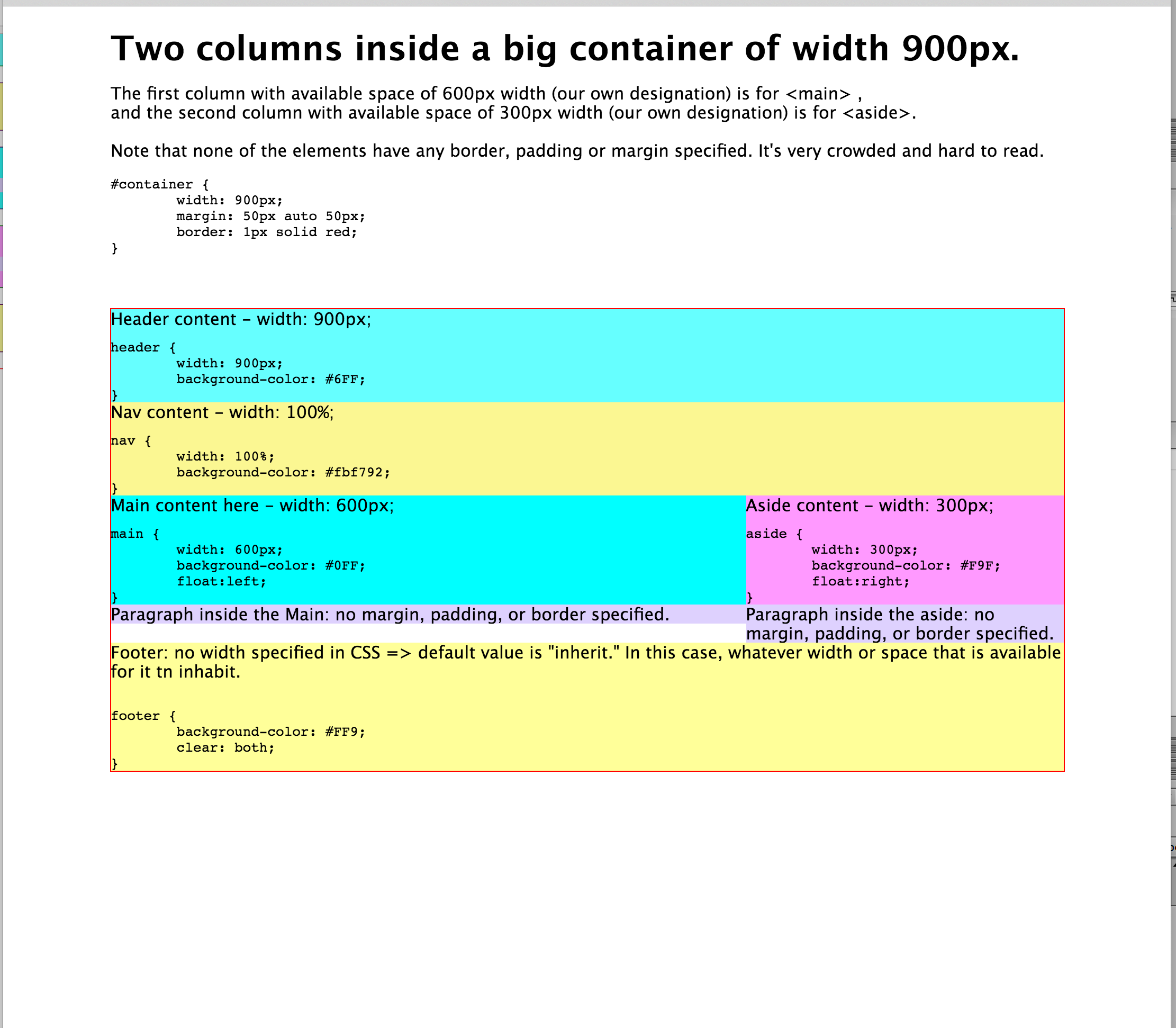
In this page, we didn’t define the width for all the elements. They occupy whatever space that is available to them and fit into the container harmoniously.



lesson6a\_2Col\_width.html

Now we are using two-column layout as an example. The concept can be applied to multiple columns.

We start with no defined border, padding or margin for all the elements inside the container.



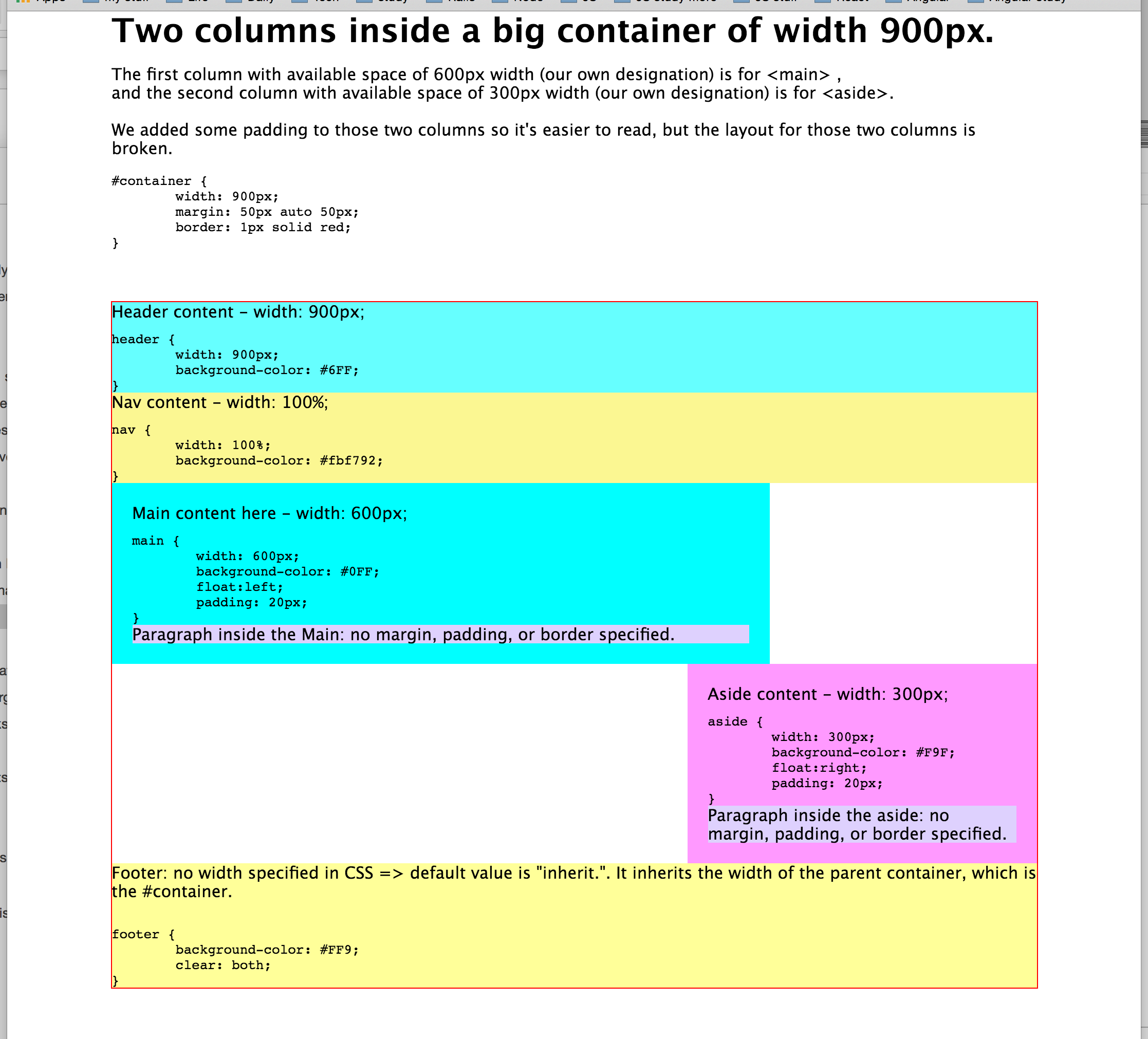
lesson6b\_2Col\_width\_padding\_broken.html

For simplicity, we only work on the two columns, <main> and <aside>. Now we add padding of 20px to both elements. Now the layout is broken. They don’t align any more.

The width we defined for <main> is 600px. That is what is called “content width,” not actual width. Actual width includes border and padding of right and left sides (top and bottom are part of the actual height). In this case, after adding 20px of padding to <main>, the actual width is now 640px.

The same happens to <aside>. The defined width is 300px. After adding 20px of padding, it now has actual width of 340px.

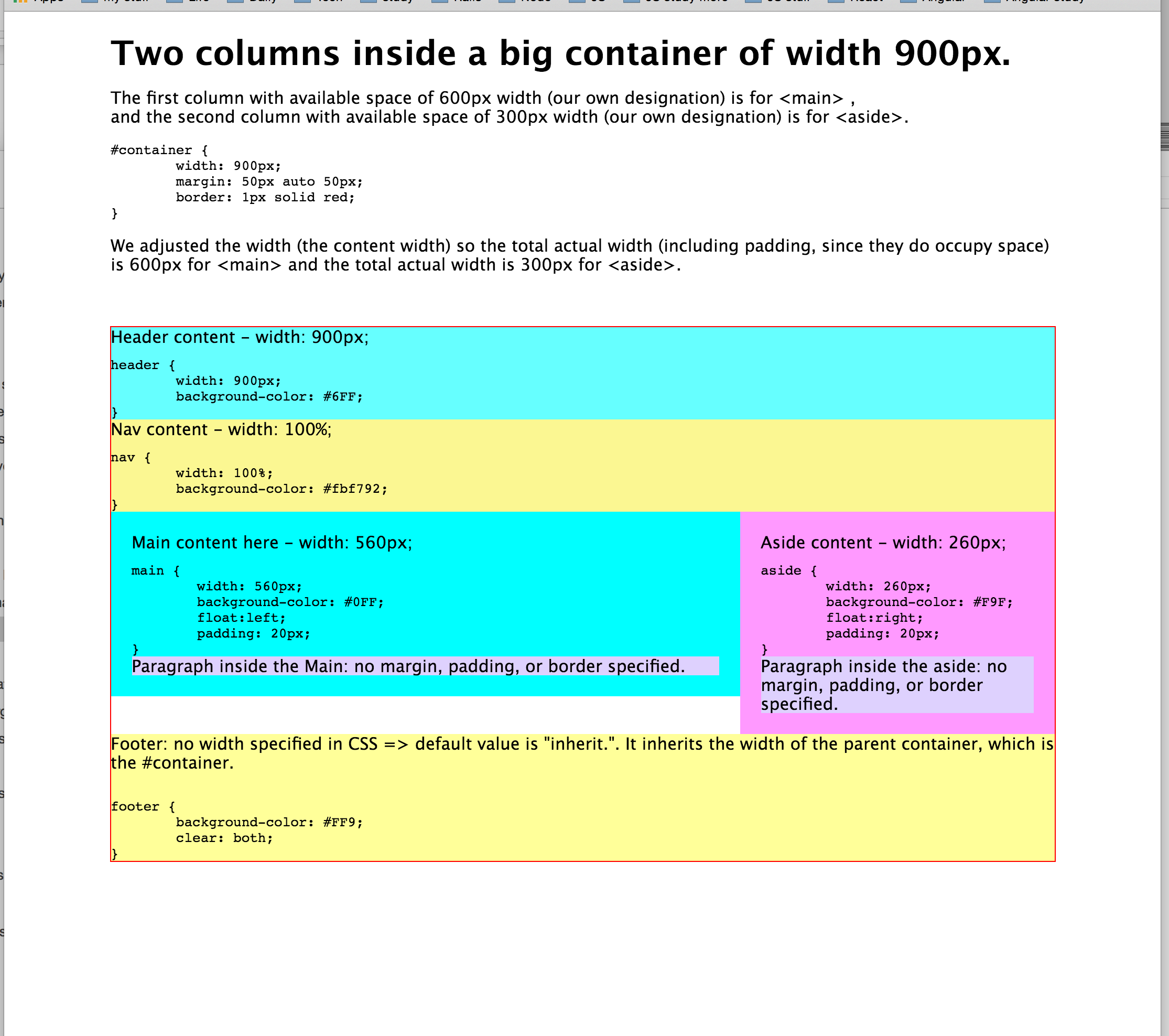
The combined width of both is now 980px. It won’t fit in the container. Since they are floating elements, no one can stick out. The <aside> gets bumped to the space below.



lesson6c\_2Col\_width\_padding\_fixed.html

To fix this, we have to adjust the width or padding for either or both elements, as long as the combined width is 900px.

In this case, we choose to reduce the defined width (the content width) for <main> to be 560px so the actual width is now 600px. We reduce the width of <aside> to 260px so the actual width now is amounted to 300px. The final combined width is now 900px. Perfect.



lesson6d\_2Col\_width\_padding\_easyFix.html

There is a new CSS property called box-sizing. It was created to ease the pain of having to recalculate the actual width of the elements every time we add padding to control spacing. It has a value border-box (box-sizing: border-box), which means it will adjust the “content width” and combine the padding and border on both left and right sides to make sure the final (actual) width matches what you declared in the “width” property.

This won’t fix the broken layout problem if you added margins that makes the combined width bigger than their container.

