I don't really like these solutions. Floats only align at their tops and need to be cleared manually. Absolute positioning takes the elements out of the flow so they do no longer affect their surroundings. And working with fixed margins and paddings immediately breaks things on the tiniest change.

But there is another player here: `vertical-align`. I think it deserves more credit. Ok, technically, using `vertical-align` for layout is a hack, since it wasn't invented for this reason. It's there to align text and elements next to text. Nonetheless, you can also use `vertical-align` in different contexts to align elements very flexible and fine-grained. The sizes of elements need not to be known. Elements stay in the flow so other elements can react to changed dimensions of those. This makes it a valuable option.

Peculiarities Of Vertical-Align

But, `vertical-align` can be a real scumbag sometimes. Working with it can be a little frustrating. There seem to be some mysterious rules at work. For example, it might happen, that the element you changed `vertical-align` for doesn't change its alignment at all, but other elements in the line do! I'm still getting dragged into the dark corners of `vertical-align` from time to time, tearing my hair.

Unfortunately, most resources on the matter are somewhat shallow. Especially, if we want to use `vertical-align` for layout. They concentrate on the misconception of trying to vertical align everything inside an element. They give basic introductions into the property and explain how elements are aligned in very simple situations. They do not explain the tricky parts.

So, I set myself the target to **clarify the behavior of vertical-align once and for all**. I ended up working through the W3C's CSS specifications and playing with some examples. The result is this article.

So, let's tackle the rules of the game.

Requirements To Use Vertical-Align

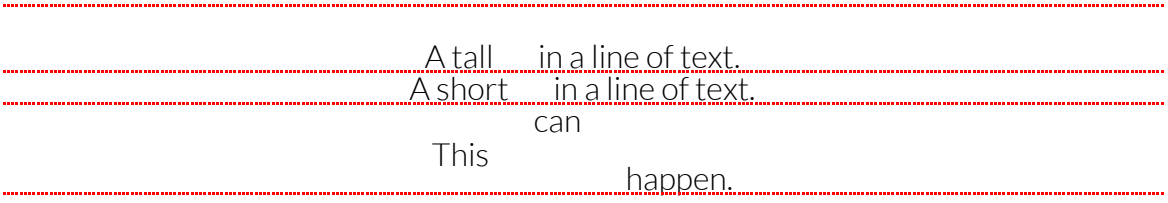
`vertical-align` is used to align inline-level elements. These are elements, whose `display` property evaluates to

- inline,
- inline-block or
- inline-table (not considered in this article).

Inline elements are basically tags wrapping text.

Inline-block elements are what their name suggests: block elements living inline. They can have a width and height (possibly defined by its own content) as well as padding, a border and margin.

Inline-level elements are laid out next to each other in lines. Once there are more elements that fit into the current line, a new line is created beneath it. All these lines have a so-called **line box**, which encloses all the content of its line. Differently sized content means line boxes of different height. In the following illustration the top and bottom of line boxes are indicated by red lines.



The line boxes trace out the field we are playing on. Inside these line boxes the property `vertical-align` is responsible for aligning the individual elements. **So, in respect to what are elements aligned?**

About Baselines and Outer Edges

The most important reference point to align vertically is the baseline of the involved elements. In some cases the top and bottom edge of the element's enclosing box becomes important, too. Let's have a look where the baseline and outer edges live for each involved type of element:

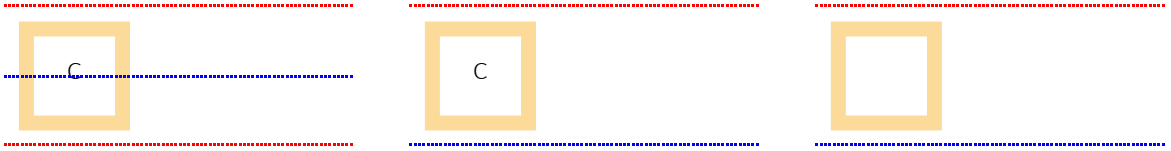
Inline Element

Here you see three lines of text next to each other. The top and bottom edge of the line height is indicated by red lines, the height of the font by green lines and the baseline by a blue line. On the left, the text has a line height set to the *same height* as the font-size. The green and red line collapsed to one line on each side. In the middle, the line height is *twice as large* as the font-size. On the right, the line height is *half as large* as the font-size.

The inline element's outer edges align themselves with the top and bottom edge of its line height. It *does not* matter, if the line height is smaller than the height of the font. So, the outer edges are the red lines in the figure above.

The inline element's baseline is the line, the characters are *sitting* on. This is the blue line in the figure. Roughly speaking, the baseline is *somewhere below the middle of the font's height*. Have look at the W3C Specs for a detailed definition.

Inline-Block Element



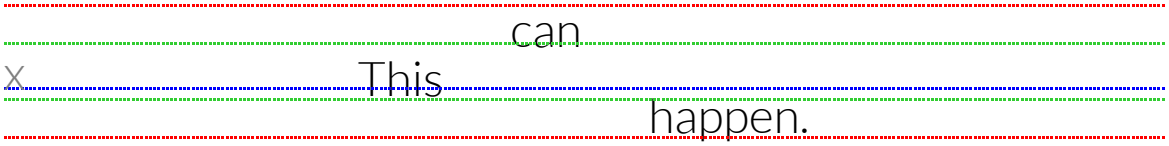
From left to right you see: an inline-block element with in-flow content (a "c"), an inline-block element with in-flow content and `overflow: hidden` and an inline-block element with *no* in-flow content (but the content area has a height). The boundaries of the margin is indicated by red lines, the border is yellow, the padding green and the content area blue. The baseline of each inline-block element is shown as a blue line.

The Inline-block element's outer edges are the top and bottom edge of its margin-box. These are the red lines in the figure.

The Inline-block element's baseline depends on whether the element has in-flow content:

- In case of in-flow content the baseline of the inline-block element is the baseline of the last content element in normal flow (example on the left). For this last element its baseline is found according to its own rules.
- In case of in-flow content but an `overflow` property evaluating to something other than `visible`, the baseline is the bottom edge of the margin-box (example in the middle). So, it is the same as the inline-block element's bottom edge.
- In case of *no* in-flow content the baseline is, again, the bottom edge of the margin-box (example on the right).

Line Box



You've already seen this setting above. This time I drew in the top and bottom of the line box's text box (green, more on this below) and the baseline (blue), too. I also highlighted the area of the text elements by giving them a grey background.

The line box has a **top edge** aligned to the top edge of the top-most element of this line and a **bottom edge** aligned to the bottom edge of the bottom-most element of the line. This is the box indicated by the red lines in the figure above.

The line box's baseline is variable:

> *"CSS 2.1 does not define the position of the line box's baseline." — the W3C Specs*

This is probably the most confusing part, when working with vertical-align. It means, the baseline is placed where ever it needs to be to fulfil all other conditions like vertical-align and minimizing the line box's height. It is a free parameter in the equation.

Since the line box's baseline is invisible, it may not immediately be obvious where it is. But, you can make it visible very easily. Just add a character at the beginning of the line in questions, like I added an "x" in the figure. If this character is not aligned in any way, it will sit on the baseline by default.

Around its baseline the line box has what we might call its **text box**. The text box can simply be thought of as an inline element inside the line box without any alignment. Its height is equal to the font-size of its parent element. Therefore, the text box only just encloses the unformatted text of the line box. The box is indicated by the green lines in the figures above. Since this text box is tied to the baseline, it moves when the baseline moves. (Side note: this text box is called strut in the W3C Specs)

Phew, this was the hard part. **Now, we know everything to put things into perspective**. Let's quickly sum up the most important facts:

- There is an area called *line box*. This is the area in which vertical alignment takes place. It has a *baseline*, a *text box* and a *top* and *bottom edge*.
- There are *inline-level elements*. These are the objects that are aligned. They have a *baseline* and and a *top* and *bottom edge*.

The Values Of Vertical-Align

By using `vertical-align` the reference points mentioned in the last sentence in the list above are set into a certain relationship.

Aligning the Element's Baseline Relative To the Line Box's Baseline

- **baseline**: The element's baseline sits exactly on top of the line box's baseline.
- **sub**: The element's baseline is shifted below the line box's baseline.
- **super**: The element's baseline is shifted above the line box's baseline.
- **<percentage>**: The element's baseline is shifted with respect to the line box's baseline by a percentage relative to the line-height.
- **<length>**: The element's baseline is shifted with respect to the line box's baseline by an absolute length.

Aligning the Element's Outer Edges Relative To the Line Box's Baseline

- **middle**: The midpoint between the element's top and bottom edge is aligned to the line box's baseline plus half of the x-height.

Aligning the Element's Outer Edges Relative To the Line Box's Text Box

One could also list these two cases under aligning relative to the line box's baseline, since the position of the text box is determined by the baseline.

- **text-top**: The element's top edge is aligned to the line box's text box top edge.
- **text-bottom**: The element's bottom edge is aligned to the line box's text box bottom edge.

Aligning the Element's Outer Edges Relative To the Line Box's Outer Edges

- **top**: The element's top edge is aligned to the line box's top edge.

- **bottom**: The element's bottom edge is aligned to the line box's bottom edge.

The formal definition is found in, of course, the W3C Specs.

Why Vertical-Align Behaves The Way It Behaves

We can now take a closer look at vertical alignment in certain situations. Especially, we will deal with situations where things might have gone wrong.

Centering an Icon

One question bugging me was the following: I have an icon I want to center next to a line of text. Just giving the icon a `vertical-align: middle` didn't seem to center it in a satisfying way. Have a look at this example:

Centered? Centered!

```
<!-- left mark-up -->
<span class="icon middle"></span>
Centered?

<!-- right mark-up -->
<span class="icon middle"></span>
<span class="middle">Centered!</span>

<style type="text/css">
 .icon   { display: inline-block;
        /* size, color, etc. */ }

 .middle { vertical-align: middle; }
</style>
```

Here is the example again, but I drew in some auxiliary lines you already know from above:

This sheds some light on our matter. Because the text on the left isn't aligned at all, it sits on the baseline. The thing is, by aligning the box with `vertical-align: middle` we are aligning it to the middle of the lower case letters without ascenders (half of the x-height). So, characters with ascenders stand out at the top.

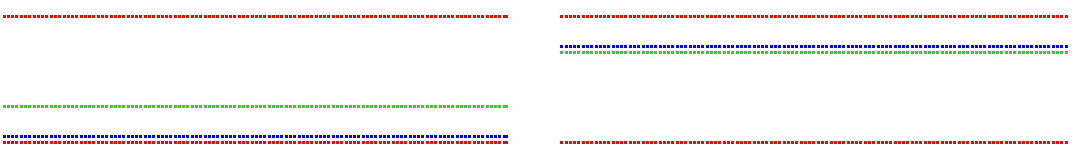
On the right, we take the whole area of the font and align its midpoint vertically, too. The text's baseline shifts slightly below the line box's baseline to achieve this. The Result is a nicely centered text next to an icon.

Movement Of the Line Box's Baseline

This is a common pitfall when working with `vertical-align`: The position of the line box's baseline is affected by all elements in that line. Let's assume, an element is aligned in such a way, that the baseline of the line box has to move. Since most vertical alignment (except top and bottom) is done relative to this baseline, this results in an adjusted position of all other elements in that line, too.

Some Examples:

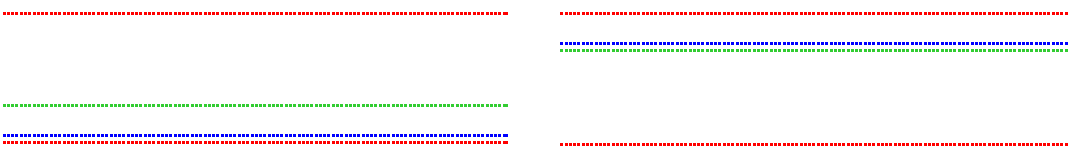
- If there is a tall element in the line spanning across the complete height, `vertical-align` has no effect on it. There is no space above its top and below its bottom, that would let it move. To fulfil its alignment relative to the line box's baseline, the line box's baseline has to move. The short box has a `vertical-align: baseline`. On the left, the tall box is aligned `text-bottom`. On the right, it is aligned `text-top`. You can see the baseline jumping up taking the short box with it.



```
<!-- left mark-up -->
<span class="tall-box text-bottom"></span>
<span class="short-box"></span>

<!-- right mark-up -->
<span class="tall-box text-top"></span>
<span class="short-box"></span>

<style type="text/css">
 .tall-box,
 .short-box   { display: inline-block;
            /* size, color, etc. */ }

 .text-bottom { vertical-align: text-bottom; }
 .text-top    { vertical-align: text-top; }
</style>
```

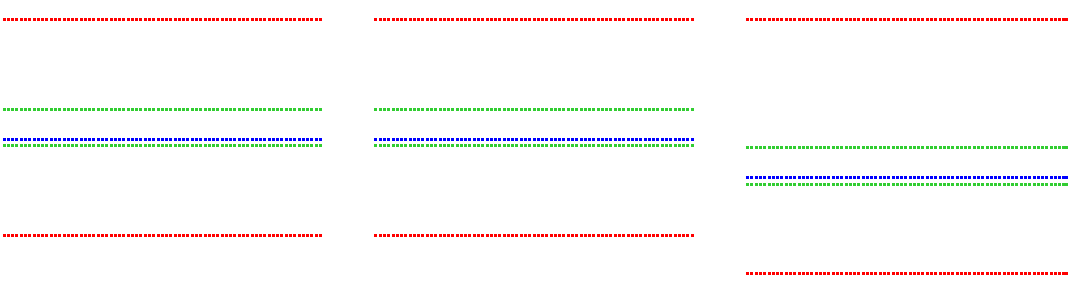
The same behaviour shows up when aligning a tall element with other values for `vertical-align`.

- Even setting `vertical-align` to `bottom` (left) and `top` (right) moves the baseline. This is strange, since the baseline shouldn't be involved at all.



```
<!-- left mark-up -->
<span class="tall-box bottom"></span>
<span class="short-box"></span>

<!-- right mark-up -->
<span class="tall-box top"></span>
<span class="short-box"></span>

<style type="text/css">
 .tall-box,
 .short-box { display: inline-block;
          /* size, color, etc. */ }

 .bottom   { vertical-align: bottom; }
 .top      { vertical-align: top; }
</style>
```

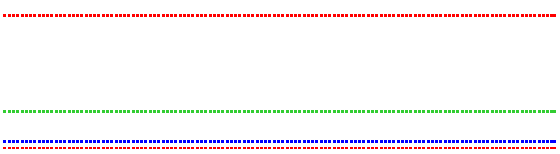
- Placing two larger elements in a line and vertically aligning them moves the baseline where it fulfils both alignments. Then the height of the line box is adjusted (left). Adding a third element, that does not go beyond the line box's edges because of its alignment, affects neither the line box's height nor the baseline's position (middle). If it *does* go beyond the line box's edges, the line box's height and baseline are adjusted, again. In this case, our first two boxes are pushed down (right).



```
<!-- left mark-up -->
<span class="tall-box text-bottom"></span>
<span class="tall-box text-top"></span>

<!-- mark-up in the middle -->
<span class="tall-box text-bottom"></span>
<span class="tall-box text-top"></span>
<span class="tall-box middle"></span>

<!-- right mark-up -->
<span class="tall-box text-bottom"></span>
<span class="tall-box text-top"></span>
<span class="tall-box text-100up"></span>

<style type="text/css">
 .tall-box   { display: inline-block;
           /* size, color, etc. */ }

 .middle     { vertical-align: middle; }
 .text-top   { vertical-align: text-top; }
 .text-bottom { vertical-align: text-bottom; }
 .text-100up  { vertical-align: 100%; }
</style>
```

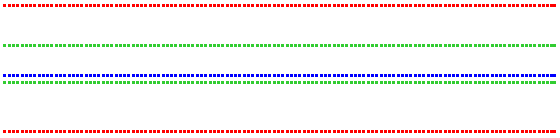
There Might Be a Little Gap Below Inline-Level Elements

Have a look at this setting. It's common if you try to vertical-align `li` elements of a list.



```
<ul>
  <li class="box"></li>
  <li class="box"></li>
  <li class="box"></li>
</ul>

<style type="text/css">
  .box { display: inline-block;
         /* size, color, etc. */ }
</style>
```

As you can see, the list items sit on the baseline. Below the baseline is some space to shelter the descenders of a text. This is causing the gap. The Solution? Just move the baseline out of the way, for example by aligning the list items with `vertical-align: middle`:



```
<ul>
  <li class="box middle"></li>
  <li class="box middle"></li>
  <li class="box middle"></li>
</ul>

<style type="text/css">
  .box    { display: inline-block;
            /* size, color, etc. */ }

  .middle { vertical-align: middle; }
</style>
```

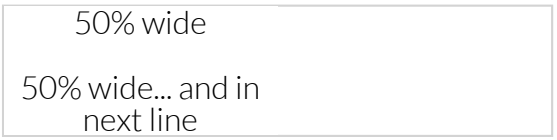
This scenario does not occur for inline-blocks having text content, since content already moves the baseline up.

A Gap Between Inline-Level Elements Is Breaking the Layout

This is mainly a problem of inline-level elements themselves. But since they are a requirement of vertical-align, it is good to know about this.

You can see this gap in the former example between the list items. The gap comes from the white-space between inline-elements present in your mark-up. All white-

space between inline-elements is collapsed into one space. This space gets in the way, if we want to place two inline elements next to each other and giving them `width: 50%`, for example. There is not enough space for two 50%-elements and a space. So the line breaks into two lines destroying the layout (left). To remove the gap, we need to remove the white-space, for example with html comments (right).




```
<!-- left mark-up -->
<div class="half">50% wide</div>
<div class="half">50% wide... and in next line</div>

<!-- right mark-up -->
  <div class="half">50% wide</div><!--
--><div class="half">50% wide</div>

<style type="text/css">
 .half { display: inline-block;
        width: 50%; }
</style>
```

Vertical-Align Demystified

Yea, that's it. It is not very complicated once you know the rules. If `vertical-align` does not behave, just ask these questions:

- Where is the baseline and top and bottom edge of the line box?
- Where is the baseline and top and bottom edge of the inline-level elements?

This will corner the solution to the problem.

by Christopher Aue
March 05, 2014

495

#inline-block
#vertical-align
#line-box
#css

293

25 Comments Christopher Aue 1 Login

Recommend 8 Share Sort by Best

Join the discussion…

Janusz Dziurzynski • a year ago

This is the best, most lucid, and thorough explanation of vertical-align I've seen so far! Thanks!

1 • Reply • Share ›

Francisc • 8 days ago

Excellent post, probably the best explanation of `vertical-align` out there.

• Reply • Share ›

wolfman • 2 months ago

Thanks Christopher I was trying to bottom align two text elements: one at the left corner the other to the right. I tried absolute positioning and floats; but I'm not too smart to make it work without any hacks. Initial search in Stack Overflow didn't yield any immediate answer. Implemented your suggestion of 50% width on each, display: inline-block. Then just have to text-align right the 2nd text. Solved my problem. Thanks!

• Reply • Share ›

Ste Cooper • 4 months ago

fan-dabi-dozi !

• Reply • Share ›

Gastón Sánchez • 5 months ago

Thank you!

• Reply • Share ›

ankeetmaini • 6 months ago

Bravo!

• Reply • Share ›

Blago • 7 months ago

I don't see any gap below when using "normal" without intrinsic dimensions inline-block elements. See here http://jsbin.com/guhoye, ? I only see gaps when using elements with intrinsic dimensions like images, form elements http://jsbin.com/jakazo http://jsbin.com/dafefi

• Reply • Share ›

Christopher Aue Author ↗ Blago • 7 months ago

Yeah, that's because the baseline of an inline-block depends on whether it has content like text or not. Have a look at: http://christopheraue.net/2014...

1 • Reply • Share ›

Marko Koron • 7 months ago

Very useful post! Solved an issue I had with this info.

• Reply • Share ›

Doc Kodam • a year ago

what if I have 2 inline-block elements of 25% each one and I want one align to left and the other to right.

• Reply • Share ›

blazicke • a year ago

Thanks! Really useful article!

• Reply • Share ›

Mate Brkić • a year ago

Nice, I really love this kind of posts! Thanks a lot.

• Reply • Share ›

Guest • a year ago

I don't understand the following:
The text box simply is an inline element inside the line box without any alignment. Its top and bottom edges are defined by the line height.

if the text box height is determined by the line height, which property determine the height of line box?

• Reply • Share ›

Christopher Aue Author → Guest • a year ago

The height of the line box is determined by its content. There is no property to set the line box's height.

* If the line is empty its only content is the text-box. In this case the height of the line box equals the height of the text-box, i.e. the line-height. This is also true, if the line only contains text.
* If the line has content, its content is vertically aligned in a way that it needs the least space. The resulting height is the height of the line box.

The line-height is the lower bound of the line box's height. The line box gets taller if its content would not fit into it otherwise.

1 • Reply • Share ›

bwbasheer%retrievalsystems.com → Christopher Aue • 4 months ago

Is there a way to defeat this behavior, i.e., to prevent a text box area from impinging on the line above or below when the character's height would cause that. In other words, is it possible to "clip" the top and/or bottom of a the text area of a given character? Not sure if this is the right question to be asking in the context of a "tall character" that does impinge when aligned up or down

• Reply • Share ›

Christopher Aue Author bwbasheer%retrievalsystems.com
• 4 months ago

Do you have a concrete example? A fiddle or codepen maybe?

• Reply • Share ›

bwbasheer%retrievalsystems.com Christopher Aue
• 4 months ago

Thanks. Here is a link to a fiddle (hopefully; never created one before) https://jsfiddle.net/0jo6fba8/.... It does not reflect the exact circumstance I had in mind (obliterating text above/below), but it does show the undesirable vertical effect on line spacing that I'd like to avoid.

• Reply • Share ›

Christopher Aue Author bwbasheer%retrievalsystems.com
• 4 months ago

The elements in the yellow highlighted boxes are aligned with position: relative and values for top or bottom. They don't use the vertical-align property. So, in this case, you can adjust the height of these boxes by explicitly setting their line-height (e.g. on the .highlight class). A value of 0.8em looks good for example.

• Reply • Share ›

bwbasheer%retrievalsystems.com Christopher Aue
• 4 months ago

Interesting. Thanks. An observation: The effect of that change is to cause shading not to extend over the lower portion of the construct, as you can see in the paragraph that uses the code the code 9830. Nevertheless, this is a great start on fixing the problem.

• Reply • Share ›

pixelwhip • a year ago

Great post! Demystifies a bunch of things for me. Thanks!

• Reply • Share ›

Reinier Kaper • a year ago

Also of note: you can vertically align elements that use "display: table-cell;" (as long as they're inside a "display: table[-row]" of course. This can be very useful, but it almost always requires some extra wrapping elements.

Also, Firefox has some issues with positioning cells and/or elements inside it (can't recall directly which one it was).

• Reply • Share ›

SelenIT Reinier Kaper • a year ago

It couldn't position things absolutely relative to cells before version 31. Now this

bug has been fixed.

• Reply • Share ›

johndubya • a year ago

Wow, incredible and in-depth information about vertical-align. Had no clue how it really worked. I feel empowered now. Thank you!

• Reply • Share ›

Costea Melniciuc • a year ago

I loved this one...
I Thinks this is the best solution for writing maintable html and removing the white line, although i use the spacing thing to remove the gaps.

• Reply • Share ›

Nicolas Hoffmann • a year ago

It works well also with display: table-cell :)

• Reply • Share ›

Image Credit: sima dimitric

BACK TO TOP