

Why not use tables for layout in HTML? [closed]

Asked 16 years, 3 months ago Modified 6 years, 1 month ago

Viewed 481k times

664

votes



As it currently stands, this question is not a good fit for our Q&A format. We expect answers to be supported by facts, references, or expertise, but this question will likely solicit debate, arguments, polling, or extended discussion. If you feel that this question can be improved and possibly reopened, [visit the help center](#) for guidance.

Closed 13 years ago.



Locked. This question and its answers are [locked](#) because the question is off-topic but has historical significance. It is not currently accepting new answers or interactions.

It seems to be the [general opinion](#) that tables should not be used for layout in HTML.

Why?

I have never (or rarely to be honest) seen good arguments for this. The usual answers are:

- It's good to [separate content from layout](#)
But this is a fallacious argument; [Cliche Thinking](#). I guess it's true that using the table element for layout has little to do with tabular data. So what? Does my boss care? Do my users care?

Perhaps me or my fellow developers who have to maintain a web page care... Is a table less maintainable? I think using a table is [easier](#) than using divs and CSS.

By the way... why is using a div or a span good separation of content from layout and a table not? Getting a good layout with only divs often requires a lot of nested divs.

- Readability of the code
I think it's the other way around. Most people understand HTML, few understand CSS.
- It's better for SEO not to use tables
Why? Can anybody show some evidence that it is? Or a statement from Google that tables are discouraged from an SEO perspective?
- Tables are slower.
An extra tbody element has to be inserted. This is peanuts for modern web browsers. Show me some benchmarks where the use of a table significantly slows down a page.
- A layout overhaul is easier without tables, see [css Zen Garden](#).

Most web sites that need an upgrade need new content (HTML) as well. Scenarios where a new version of a web site only needs a new CSS file are not very likely. Zen Garden is a nice web site, but a bit theoretical. Not to mention its [misuse](#) of CSS.

I am really interested in good arguments to use divs + CSS instead of tables.

html

css

Share

edited Nov 6, 2018 at 0:07

community wiki

21 revs, 14 users 69%

[Benno Richters](#)

Agreed, tables are fine when presenting tabular data. They should be avoided when using it purely for layout. Then again, sometimes, you have to take the easy road now and improve it later. Just view source and you'll see what I mean.

– [Haacked](#) Sep 17, 2008 at 16:07

There is a duplicate Q and A at

stackoverflow.com/questions/30251/tables-instead-of-divs

– [Onorio Catenacci](#) Sep 17, 2008 at 18:30

-
- 11** The answer is simple: it depends. If tables are used to solve a specific problem that current CSS versions can't, they are well used. If you start getting tables inside tables, inside millions of tables then you're doing it wrong. If it's the occasional table just

to layout some 2 columns or something like that, I don't disallow it on my team: it's faster and easier to do it. (Myself, I always try to use CSS, but at the end of the day, delivery is more important than correct semantic HTML) – [AlfaTeK](#) Jan 21, 2010 at 14:48

- 1 @Camilo SO still lives in the 20th century. Jeff apparently does not know how to use the `ul` tag. Have a look at all of the lists on this site (badges, related questions, recent tags). They're all either single columns or long paragraphs separated with `br`. – [Yi Jiang](#) Sep 20, 2010 at 12:50
- 2 @Brad: Depending on some specific details (that's my get out for any clever comebacks ;-)) that usage of DIVs is STILL better than abusing tables. Two parts of a document that happen to be laid out alongside each other can legitimately be contained in DIV elements, but they're certainly NOT tabular data. It doesn't matter what one specific styling happens to be; the content is either tabular or not. Note: I am NOT advocating DIVitis either :) – [Bobby Jack](#) Nov 1, 2010 at 18:20

Comments disabled on deleted / locked posts / reviews |

66 Answers

Sorted by:

Highest score (default)



1

2

3

Next

495

votes

I'm going to go through your arguments one after another and try to show the errors in them.



It's good to separate content from layout But this is a fallacious argument; Cliché Thinking.

It's not fallacious at all because HTML was designed intentionally. Misuse of an element might not be

completely out of question (after all, new idioms have developed in other languages, as well) but possible negative implications have to be counterbalanced. Additionally, even if there were no arguments against misusing the `<table>` element today, there might be tomorrow because of the way browser vendors apply special treatment to the element. After all, they know that “`<table>` elements are for tabular data only” and might use this fact to improve the rendering engine, in the process subtly changing how `<table>`s behave, and thus breaking cases where it was previously misused.

So what? Does my boss care? Do my users care?

Depends. Is your boss pointy-haired? Then he might not care. If she's competent, then she will care, because the users [will](#).

Perhaps me or my fellow developers who have to maintain a web page care... Is a table less maintainable? I think using a table is easier than using divs and css.

~~The majority of professional web developers seem to oppose you~~^[citation needed]. That tables are in fact less maintainable should be obvious. Using tables for layout means that changing the corporate layout will in fact mean changing every single page. This can be very expensive. On the other hand, judicious use of semantically

meaningful HTML combined with CSS *might* confine such changes to the CSS and the pictures used.

By the way... why is using a div or a span good separation of content from layout and a table not? Getting a good layout with only divs often requires a lot of nested divs.

Deeply nested `<div>`s are an anti-pattern just as table layouts. Good web designers don't need many of them. On the other hand, even such deep-nested divs don't have many of the problems of table layouts. In fact, they can even contribute to a semantic structure by logically dividing the content in parts.

Readability of the code I think it's the other way around. Most people understand html, little understand css. It's simpler.

“Most people” don't matter. Professionals matter. For professionals, table layouts create many more problems than HTML + CSS. This is like saying I shouldn't use GVim or Emacs because Notepad is simpler for most people. Or that I shouldn't use LaTeX because MS Word is simpler for most people.

It's better for SEO not to use tables

I don't know if this is true and wouldn't use this as an argument but it would be logical. Search engines search for *relevant* data. While tabular data could of course be relevant, it's rarely what users search for. Users search for terms used in the page title or similarly prominent positions. It would therefore be logical to exclude tabular content from filtering and thus cutting the processing time (and costs!) by a large factor.

Tables are slower. An extra tbody element has to be inserted. This is peanuts for modern web browsers.

The extra element has got nothing to do with tables being slower. On the other hand, the layout algorithm for tables is much harder, the browser often has to wait for the whole table to load before it can begin to layout the content. Additionally, caching of the layout won't work (CSS can easily be cached). All this has been mentioned before.

Show me some benchmarks where the use of a table significantly slows down a page.

Unfortunately, I don't have any benchmark data. I would be interested in it myself because it's right that this argument lacks a certain scientific rigour.

Most web sites that need an upgrade need new content (html) as well. Scenarios where a new

version of a web site only needs a new css file
are not very likely.

Not at all. I've worked on several cases where changing the design was simplified by a separation of content and design. It's often still necessary to change some HTML code but the changes will always be much more confined. Additionally, design changes must on occasion be made dynamically. Consider template engines such as the one used by the WordPress blogging system. Table layouts would literally kill this system. I've worked on a similar case for a commercial software. Being able to change the design without changing the HTML code was one of the business requirements.

Another thing. Table layout makes automated parsing of websites (screen scraping) much harder. This might sound trivial because, after all, who does it? I was surprised myself. Screen scraping can help a lot if the service in question doesn't offer a WebService alternative to access its data. I'm working in bioinformatics where this is a sad reality. Modern web techniques and WebServices have not reached most developers and often, screen scraping is the only way to automate the process of getting data. No wonder that many biologists still perform such tasks manually. For thousands of data sets.

Share

edited Aug 1, 2010 at 18:42

- 139 "changing the corporate layout will in fact mean changing every single page" - Do people still actually duplicate the corporate layout on every page? It's so easy to resolve with master pages or user controls in .net, include files in php or classic asp, etc ... Anybody who copies the company layout like this deserves an a** kicking! ;-) – [John MacIntyre](#) May 20, 2009 at 14:30
-
- 43 Sorry but this is really "pie int he sky" wishful thinking. Users care? No. Noone cares except a small number of misguided revisionists. HTML (including tables) is far older than the relatively new notion of "semantics vs layout". Oh and source please for "the majority of professional web developers oppose you". – [cletus](#) Jun 17, 2009 at 21:48
-
- 20 Typo above: semantics were implied *from* the beginning. <table> in HTML was always only meant for tabular data, never for layouting (back in the early years you couldn't change the table looks anyway, thus preventing its use as a layout anchor). In fact, early drafts of HTML had no notion of layout at all, and HTML was never meant for layout, but for structuring text. Even more: the very first proposal of HTML repeatedly warns against abusing tags to influence the layout, and cautions to use logical over physical markup. – [Konrad Rudolph](#) Jun 18, 2009 at 9:11
-
- 109 Get a screen reader and have it read a page with a table layout. that is all. – [corymathews](#) Jul 26, 2009 at 16:42
-
- 8 @Sergio: please do not edit out relevant information. This *is* an unsourced dubious claim I'm using there and I want to make that quite clear. Your edit put a claim in my mouth that I can't back up, effectively making me a liar if this turns out to be false. – [Konrad Rudolph](#) Aug 1, 2010 at 18:49
-

288

Here's my *programmer's answer* from a [simliar thread](#)

votes



Semantics 101

First take a look at this code and think about what's wrong here...

```
class car {  
    int wheels = 4;  
    string engine;  
}  
  
car mybike = new car();  
mybike.wheels = 2;  
mybike.engine = null;
```

The problem, of course, is that a bike is not a car. The car class is an inappropriate class for the bike instance. The code is error-free, but is *semantically* incorrect. It reflects poorly on the programmer.

Semantics 102

Now apply this to document markup. If your document needs to present tabular data, then the appropriate tag would be `<table>`. If you place navigation into a table however, then you're misusing the intended purpose of the `<table>` element. In the second case, you're not presenting tabular data -- you're (mis)using the `<table>` element to achieve a presentational goal.

Conclusion

Will visitors notice? No. Does your boss care? Maybe. Do we sometimes cut corners as programmers? Sure. But should we? No. Who benefits if you use semantic markup? You -- and your professional reputation. Now go and do the right thing.

Share

edited May 23, 2017 at 11:54

community wiki

4 revs, 3 users 88%

Carl Camera

-
- 39 Sorry, that doesn't hold water. You don't use car for mybike because you would define a "bike" class instead. You can't define something else for "<table>" because it is more than a simple semantic tag -- it tells the browser how to render its content as well. – [Jimmy](#) Sep 17, 2008 at 20:40
-
- 57 Nice analogy, and great conclusion. Why should anyone have to be forced by one's boss and/or users into doing the right thing? Doesn't anyone take pride in their own work any more? – [Sherm Pendley](#) Nov 13, 2008 at 9:10
-
- 39 I think this is quite an arrogant post which doesn't explain anything but just repeats the same claims again without answering the question. I don't understand why it got so many upvotes???? – [markus](#) Nov 16, 2008 at 1:09
-
- 10 I agree with tharkum. This response is rather subjective, and does not actually answer the question posed. While I agree that DIVs should be used for page layout, I cannot imagine that any web designer would be confused by a table-based layout. – [Richard Ev](#) Nov 27, 2008 at 11:33
-

16 @tharkun & Richard: How come "semantically incorrect" is subjective and does not explain anything? – [Vinko Vrsalovic](#)
Dec 14, 2008 at 17:06

103

votes



Obvious answer: See [CSS Zen Garden](#). If you tell me that you can easily do the same with a table-based layout (remember - the HTML isn't changing) then by all means use tables for layout.

Two other important things are accessibility and SEO.

Both care about in what order information is presented.

You cannot easily present your navigation at the top of the page if your table-based layout puts it in the 3rd cell of the 2nd row of the 2nd nested table on the page.

So your answers are maintainability, accessibility and SEO.

Don't be lazy. Do things the right and proper way even if they are a bit harder to learn.

Share

answered [Sep 17, 2008 at 13:33](#)

community wiki
[erlando](#)

24 An often used trick in Zen Garden is replacing text by images, and define that in the CSS, thus making it invisible from HTML. Very, very wrong. – [GvS](#) Sep 17, 2008 at 14:25

3 I'm sorry but that's not right. The text is still in the HTML - that's one of the requirements of a CSSZG design. The HTML has to remain unchanged. – [erlando](#) Sep 17, 2008 at 14:41

10 If you look closely at some of the designs, the text in some headers is different from the text in the HTML. That's because the HTML is made invisible, and instead an image is inserted. (Example: csszengarden.com/?cssfile=/212/212.css&page=0, The ?Path? to ?Achievement?) – [GvS](#) Sep 17, 2008 at 15:31

6 Sorry, there's absolutely no evidence div layouts are better for SEO. Also, Google themselves have stated that HTML validation doesn't matter to them - a slightly different issue but one aimed towards tables because they rarely validate. – [DisgruntledGoat](#) Jul 4, 2009 at 23:39

“Most of you are familiar with the virtues of a programmer. There are three, of course: laziness, impatience, and hubris.”
- Larry Wall – [inkredibl](#) Dec 22, 2010 at 9:54

91 [See this duplicate question.](#)

votes



One item you're forgetting there is accessibility. Table-based layouts don't translate as well if you need to use a screen reader, for example. And if you do work for the government, supporting accessible browsers like screen readers may be *required*.

I also think you underestimate the impact of some of the things you mentioned in the question. For example, if you are both the designer and the programmer, you may not have a full appreciation of how well it separates presentation from content. But once you get into a shop

where they are two distinct roles the advantages start to become clearer.

If you know what you're doing and have good tools, CSS really does have significant advantages over tables for layout. And while each item by itself may not justify abandoning tables, taken together it's generally worth it.

Share

edited May 23, 2017 at 12:10

community wiki

4 revs

Joel Coehoorn

Yes... indeed. I see that's duplicate. I missed it. Any action required besides promising I will be more careful next time :-) ?


– Benno Richters Sep 17, 2008 at 13:30

Don't worry. This will be more and more common as the number of questions goes up. I didn't even downvote. We just want to be sure that as much as possible they eventually all point to a common source. – Joel Coehoorn Sep 17, 2008 at 13:53

8 I really like this answer. Using semantically meaningful tags isn't just a matter of tradition, it allows those obscure non-browsers (screen readers, screen scrapers, various parsers) to correctly categorize the various objects on your page.
– Phantom Watson Jan 9, 2009 at 17:01

6 I was hoping someone would mention this. Accessibility should be a top priority! – Andrew May 12, 2009 at 9:01

I can't say I agree with the notion that accessibility should be a top priority in a commercial exercise. The cost benefit ratio is

not feasible. It's like putting a wheelchair ramp on a mountain-climbing shop - a ridiculous waste of resources that could be applied to items with better CBR. If you were talking about a medical facility then a wheelchair ramp would be a priority. Likewise, I would only bother with web page accessibility for sites pitched at vision impaired people. – [Peter Wone](#) Jun 15, 2011 at 0:27 

54
votes



Unfortunately, CSS Zen Garden can no longer be used as an example of good HTML/CSS design. Virtually all of their recent designs use graphics for section heading. These graphic files are specified in the CSS.

Hence, a website whose purpose is to show the advantage of keeping design out of content, now regularly commits the UNSPEAKABLE SIN of putting content into design. (If the section heading in the HTML file were to change, the section heading displayed would not).

Which only goes to show that even those advocate the strict DIV & CSS religion, can't follow their own rules. You may use that as a guideline in how closely you follow them.

Share

answered [Sep 17, 2008 at 14:09](#)

community wiki
[James Curran](#)

18 You mean they're doing `<h1>Some Text</h1>` and then in their css: `h1 { background-image('foo.jpg'); text-indent: -3000px }` ? This is the *correct* way of doing it

because you're retaining maximum semantic information in the style-less html. Or maybe I misunderstood you. – [Karan](#) Oct 12, 2008 at 1:08

9 Karen's right, you're wrong, James. Your example is a straw man. To forget to change the image when you changed the semantic HTML would be stupid. So is leaving your laptop on a bus. What's your point? – [AmbroseChapel](#) Mar 27, 2009 at 9:31

36 @Ambrose: Because the entire point of the friggin' site is to demonstrate the separation of content & style. Content & Style should properly be handled by different people, neither of whom should have to "remember" what the other changed. – [James Curran](#) Mar 27, 2009 at 14:38

5 Mr S&N: that would make matters even worse. You would have to dynamically generate new images -- in the correct style. So now you've moved styling from CSS to business layer code. – [James Curran](#) Mar 27, 2009 at 14:46

9 @James: Content and style can't always be handled by different people. When the content is stylized, collaboration must occur. How is `<h1></h1>` any more maintainable than `<h1 class="something image">Something</h1>` ? In either example something.png needs to be updated. But the second example is far more accessible. – [Josh](#) May 1, 2010 at 22:01

48
votes



This isn't the definitive argument, by any means, but with CSS you can take the same markup and change the layout depending on medium, which is a nice advantage. For a print page you can quietly suppress navigation without having to create a printer-friendly page, for example.

community wiki
[expedient](#)

Good point, though you can use css to hide cells in a table based layout, to supress navigation in a printable version for instance, a CSS layout gives you much more flexibility behind merely hiding data. – [Cory House](#) Oct 4, 2009 at 2:26

I hit this with one of my applications; boy was I happy when I realized how easy it was to create the "printer friendly" version with just some print CSS for the same pages as were on the screen. – [L. Cornelius Dol](#) May 28, 2011 at 7:36

45

votes



One table for layout wouldn't be that bad. But you can't get the layout you need with just one table most of the time.

Pretty soon you have 2 or three nested tables. This becomes very cumbersome.

- It IS a LOT harder to read. That's not up to opinion. There's just more nested tags with no identifying marks on them.
- Separating content from presentation is a good thing because it allows you to focus on what you're doing. Mixing the two leads to bloated pages that are hard to read.
- CSS for styles allows your browser to cache the files and subsequent requests are much faster. This is HUGE.

- Tables lock you into a design. Sure, not everyone needs the flexibility of CSS Zen Garden, but I've never worked on a site where I didn't need to change the design a little bit here and there. It's much easier with CSS.
- Tables are hard to style. You don't have very much flexibility with them (i.e. you still need to add HTML attributes to fully control a table's styles)

I haven't used tables for non-tabular data in probably 4 years. I haven't looked back.

I'd really like to suggest reading [CSS Mastery](#) by Andy Budd. It's fantastic.

[Image at ecx.images-amazon.com http://ecx.images-amazon.com/images/I/41TH5NFKPEL._SL500_BO2,204,203,200_Plsitb-dp-500-arrow,TopRight,45,-64_OU01_AA240_SH20_.jpg](http://ecx.images-amazon.com/images/I/41TH5NFKPEL._SL500_BO2,204,203,200_Plsitb-dp-500-arrow,TopRight,45,-64_OU01_AA240_SH20_.jpg)

Share

edited May 1, 2010 at 21:48

community wiki
2 revs, 2 users 95%
[Ben Scheirman](#)

Contains good examples for box model, selectors, and positioning. – [Kamran Khan](#) Jan 14, 2010 at 12:41

36

votes



It's good to separate content from layout

But this is a fallacious argument; Cliche Thinking

It's a fallacious argument because HTML tables are layout! The content is the *data* in the table, the presentation is the table itself. This is why separating CSS from HTML can be very difficult at times. You're not separating content from presentation, you're separating presentation from presentation! A pile of nested divs is no different than a table - it's just a different set of tags.

The other problem with separating the HTML from the CSS is that they need intimate knowledge of one another - you really can't separate them fully. The tag layout in the HTML is tightly coupled with the CSS file no matter what you do.

I think tables vs divs comes down to the needs of your application.

In the application we develop at work, we needed a page layout where the pieces would dynamically size themselves to their content. I spent days trying to get this to work cross-browser with CSS and DIVs and it was a complete nightmare. We switched to tables and it all just *worked*.

However, we have a very closed audience for our product (we sell a piece of hardware with a web interface) and accessibility issues are not a concern for us. I don't know why screen readers can't deal with tables well, but I guess if that's the way it is then developers have to handle it.

Share

edited Nov 14, 2011 at 20:18

community wiki

3 revs, 2 users 96%

17 of 26

6 screenreaders deals with tables ok.. It's the way that tables don't deal with screenreaders that's the problem. A table-based layout is prone to present the information in an inaccessible manner. Think about how a right-side navigation would look like. It would be pretty far down the page. – [erlando](#) Sep 17, 2008 at 14:08

Separation of content from presentation only occurs when the markup is semantic. Most people don't know how to write really semantic mark up, and they also can't present that markup properly with css. That is a failure of the dev, not the technology. I can do it, it's not that hard. – [Pete Michaud](#) Sep 17, 2008 at 17:33

8 Just because <table> or <div> have a *default* presentation in most screen agents, doesn't equate them to being presentation elements instead of semantic elements. – [Mark Brackett](#) Sep 18, 2008 at 10:34

1 I'm not talking about HTML specifically, I'm talking about conceptually in basic UI design. A table dictates how things are laid out on the screen, i.e. how they are presented to the user. That is presentation. – [17 of 26](#) Sep 19, 2008 at 2:46

1 "I spent days trying to get this to work" - if something I'm trying to figure out ever takes more than a couple of hours, I put it to the StackOverflow community. Not knowing how to do something correctly is no excuse for not doing it correctly. Especially when we have all the answers at our fingertips. – [DaveDev](#) Aug 6, 2010 at 23:43

23

votes



CSS/DIV - it's just jobs for the design boys, isn't it. The hundreds of hours I've spent debugging DIV/CSS issues, searching the Internet to get some part of markup working with an obscure browser - it drives me mad. You make one



little change and the whole layout goes horrendously wrong - where on earth is the logic in that. Spending hours moving something 3 pixels this way then something else 2 pixels the other to get them all to line up. This just seems plain wrong to me somehow. Just because you're a purist and something is "not the right thing to do" doesn't mean you should make use of it to the nth degree and under all circumstances, especially if it makes your life 1000 times easier.

So I've finally decided, purely on commercial grounds, although I keep use to minimum, if I anticipate 20 hours work to get a DIV placed correctly, I'll stick in a table. It's wrong, it upsets the purists, but in most cases it costs less time and is cheaper to manage. I can then concentrate on getting the application working as the customer wants, rather than pleasing the purists. They do pay the bills after all and my argument to a manager enforcing the use of CSS/DIV - I would merely point out the customers pay his salary as well!

The only reason all these CSS/DIV arguments occur is because of the shortcoming of CSS in the first place and because the browsers aren't compatible with each other and if they were, half the web designers in the world would be out of a job.

When you design a windows form you don't try moving controls around after you have laid them out so I kind of think it's strange to me why you would you want to do this with a web form. I simply can't understand this logic. Get

the layout right to start with and what's the problem. I think it's because designers like to flirt with creativity, whilst application developers are more concerned with actually getting the application working, creating business objects, implementing business rules, working out how bits of customer data relates to each other, ensuring the thing meets the customers requirements - you know - like the real world stuff.

Don't get me wrong, both arguments are valid, but please don't criticise developers for choosing an easier, more logical approach to designing forms. We often have more important things to worry about than the correct semantics of using a table over a div.

Case in point - based on this discussion I converted a few existing tds and trs to divs. 45 minutes messing about with it trying to get everything to line up next to each other and I gave up. TDs back in 10 seconds later - works - straight away - on all browsers, nothing more to do. Please try to make me understand - what possible justification do you have for wanting me to do it any other way!

Share

[edited Dec 13, 2010 at 17:27](#)

community wiki
[4 revs, 3 users 53%](#)
[Peter Mortensen](#)

I couldn't agree more with this post. In the 10 years I've been in design, I cannot think of a single time where the argument "we

use CSS because it makes sitewide changes easier to manage" played out as accurate. – [Daniel Szabo](#) Nov 25, 2010 at 22:20

- 6 know what makes sitewide changes easy to manage? **MVC** and **template systems** – [Jiaaro](#) Nov 29, 2010 at 18:32
-

Please don't get me wrong - I still use CSS but I use it to apply style (colour, background images and so forth) and not layout. CSS seems to be pretty much consistent across browsers when used to control style only. Where it is flawed and falls down in a big way is the way in which layout is both specified and implemented across browsers. Has anyone ever tried to get ASP.NET MasterPages working reliably with DIVs? Its almost impossible! – [Tim Black](#) Feb 11, 2011 at 22:19

21
votes



Layout should be easy. The fact that there are articles written on how to achieve a dynamic three column layout with header and footer in CSS shows that it is a poor layout system. Of course you can get it to work, but there are literally hundreds of articles online about how to do it. There are pretty much no such articles for a similar layout with tables because it's patently obvious. No matter what you say against tables and in favor of CSS, this one fact undoes it all: a basic three column layout in CSS is often called "The Holy Grail".

If that doesn't make you say "WTF" then you really need to put down the kool-aid now.

I love CSS. It offers amazing styling options and some cool positioning tools, but as a layout engine it is deficient. There needs to be some type of dynamic grid positioning

system. A straightforward way to align boxes on multiple axis without knowing their sizes first. I don't give a damn if you call it `<table>` or `<gridlayout>` or whatever, but this is a basic layout feature that is missing from CSS.


The larger problem is that by not admitting there are missing features, the CSS zealots have been holding CSS back from all it could be. I'd be perfectly happy to stop using tables if CSS provided decent multi-axis grid positioning like basically every other layout engine in the world. (You do realize this problem has already been solved many times in many languages by everyone except the W3C, right? And nobody else denied that such a feature was useful.)

Sigh. Enough venting. Go ahead and stick your head back in the sand.

Share

[edited Nov 1, 2011 at 23:56](#)

community wiki
[2 revs, 2 users 94%](#)
[needlestack](#)

"CSS zealots" (as you put it) are not ignorant of this fact. The next CSS spec has not just one, but **two** solutions to fix the problems with layouts in CSS. Template Layouts: w3.org/TR/css3-layout And Grid Positioning: w3.org/TR/css3-grid This doesn't introduce competing specs. The 2 specs actually complement each other. As of the time of writing this comment though, no browser implements it, yet. – [Dan Herbert](#) Feb 11, 2011 at 4:58 

+1: I completely agree. You shouldn't have to "get it to work" (although I don't like tables either). – [3lectrologos](#) Feb 11, 2011 at 17:30

This is 100% exactly how I feel. I wish I could upvote you to the top answer. – [bobwienholt](#) Feb 2, 2012 at 14:55

19
votes



According to 508 compliance (for screen readers for visually impaired), tables should only be used to hold data and not for layout as it causes the screen readers to freak out. Or so I've been told.

If you assign names to each of the divs, you can skin them all together using CSS as well. They're just a bit more of a pain to get to sit the way you need them to.

Share

answered [Sep 17, 2008 at 13:21](#)

community wiki
[Rob](#)

18 Here's a section of html from a recent project:

votes



```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en"
<head>
  <title>{DYNAMIC(TITLE)}</title>
  <meta http-equiv="content-type" content="text/html;ch
  <meta http-equiv="Content-Style-Type" content="text/
  <link rel="stylesheet" type="text/css" href="./style
</head>
<body>
  <div id="header">
    <h1><!-- Page title --></h1>
    <ol id="navigation">
      <!-- Navigation items -->
    </ol>
    <div class="clearfix"></div>
  </div>
  <div id="sidebar">
    <!-- Sidebar content -->
  </div>
  <!-- Page content -->
  <p id="footer"><!-- Footer content --></p>
</body>
</html>
```

And here's that same code as a table based layout.

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en"
<head>
  <title>{DYNAMIC(TITLE)}</title>
  <meta http-equiv="content-type" content="text/html;ch
  <meta http-equiv="Content-Style-Type" content="text/
  <link rel="stylesheet" type="text/css" href="./style
</head>
```

```

<body>
  <table cellpadding="0">
    <tr>
      <td><!-- Page Title --></td>
      <td>
        <table>
          <tr>
            <td>Navitem</td>
            <td>Navitem</td>
          </tr>
        </table>
      </td>
    </tr>
  </table>

  <table>
    <tr>
      <td><!-- Page content --></td>
      <td><!-- Sidebar content --></td>
    </tr>
    <tr>
      <td colspan="2">Footer</td>
    </tr>
  </table>
</body>
</html>

```



The only cleanliness I see in that table based layout is the fact that I'm overzealous with my indentation. I'm sure that the content section would have a further two embedded tables.

Another thing to think about: **filesizes**. I've found that table-based layouts are twice the size of their CSS counterparts usually. On our high-speed broadband that isn't a huge issue but it is on those with dial up modems.

community wiki
2 revs, 2 users 97%
Ross

-
- 3 speed isn't the only thing that is hurt by larger files: many companies pay for bandwidth. If you can cut your client's bandwidth bills in half simply by coding well, that's a great advantage. – [Mr. Shiny and New 安宇](#) Oct 28, 2008 at 13:30
-
- 2 Yeah but don't forget that while the HTML might be twice as large in a CSS-less layout, using a DIV+CSS layout will result in (at least) an extra HTTP Request for the CSS file, plus the bandwidth usage for the file itself. – [goldenratio](#) May 8, 2009 at 22:35
-
- 12 But the css file need only be downloaded once, where as the whole table structure is resent on each page request.
– [user151841](#) Dec 17, 2009 at 16:37
-
- 3 You've picked a design well-suited to div+css and shown that it's more verbose in a table-based design? So what: It would be just as easy to create a design that was well-suited to table-based layout and show the hoops you'd have to jump through to replicate it in CSS. – [Draemon](#) Jan 29, 2010 at 20:12
-
- 4 @Draemon: Maybe you'd like to give an example?
– [Bobby Jack](#) Nov 8, 2010 at 12:19
-

14 votes

I'd like to add that div-based layouts are easier to maintain, evolve, and refactor. Just some changes in the CSS to reorder elements and it is done. From my experience, redesign a layout that uses tables is a nightmare (more if there are nested tables).

Your code also has a meaning from a [semantic](#) point of view.

Share

answered [Sep 17, 2008 at 13:28](#)

community wiki
[Guido](#)

My dream is having a graphic designer which takes the objects/data I provide and put them in a page without I having to care about CSS, DIV, TABLE... :) – [Manrico Corazzi](#) Sep 17, 2008 at 15:57

I second that Manrico - a visual designer that would allow you to construct a layout and define fixed and percentage dimensions and then generate minimal HTML and CSS would be exceedingly useful! – [Richard Ev](#) Nov 27, 2008 at 11:35

The problem I have with the semantic web concept is that in HTML 4 the vocabulary is very limited and designed for primarily technical documents. Many sites with commercial/marketing goals may have elements which do not fit neatly into this vocabulary. HTML 5 fixes some of this with tags like nav, header, footer but for now we are stuck using tags like span which actually say very little about how the content should be interpreted. – [Nick Van Brunt](#) May 28, 2010 at 16:29



13 No arguments in DIVs favour from me.

votes

I'd say : If the shoe fits, wear it.





It's worth noting that it's difficult if not impossible to find a good DIV+CSS method of rendering contents in two or three columns, that is consistent on all browsers, and still looks just the way I intended.

This tips the balance a bit towards tables in most of my layouts, and although I feel guilty of using them (dunny why, people just say it's bad so I try to listen to them), in the end , the pragmatic view is it's just easier and faster for me to use TABLEs. I'm not being payed by the hour, so tables are cheaper for me.

Share

answered [Sep 17, 2008 at 13:48](#)

community wiki
[Radu094](#)

-
- 1 if you want to create a two or three column website with divs+css that works in all browsers, you should by all means do not start from scratch. There are many barebone templates available from the web which you can use as a base. Those are usually optimized to display correctly in all browsers ie6+
– [arturh](#) [Sep 17, 2008 at 14:58](#)
-

13
votes



CSS layouts are generally much better for accessibility, provided the content comes in a natural order and makes sense without a stylesheet. And it's not just screen readers that struggle with table-based layouts: they also make it much harder for mobile browsers to render a page properly.

Also, with a div-based layout you can very easily do cool things with a print stylesheet such as excluding headers, footers and navigation from printed pages - I think it would be impossible, or at least much more difficult, to do that with a table-based layout.

If you're doubting that separation of content from layout is easier with divs than with tables, take a look at the div-based HTML at [CSS Zen Garden](#), see how changing the stylesheets can drastically change the layout, and think about whether you could achieve the same variety of layouts if the HTML was table based... If you're doing a table-based layout, you're unlikely to be using CSS to control all the spacing and padding in the cells (if you were, you'd almost certainly find it easier to use floating divs etc. in the first place). Without using CSS to control all that, and because of the fact that tables specify the left-to-right and top-to bottom order of things in the HTML, tables tend to mean that your layout becomes very much fixed in the HTML.

Realistically I think it's very hard to completely change the layout of a div-and-CSS-based design without changing the divs a bit. However, with a div-and-CSS-based layout it's much easier to tweak things like the spacing between various blocks, and their relative sizes.

Share

edited Sep 17, 2008 at 14:19

13

votes



The fact that this is a hotly debated question is a testament to the failure of the W3C to anticipate the diversity of layout designs which would be attempted. Using divs+css for semantically-friendly layout is a great concept, but the details of implementation are so flawed that they actually limit creative freedom.

I attempted to switch one of our company's sites from tables to divs, and it was such a headache that I totally scrapped the hours of work I had poured into it and went back to tables. Trying to wrestle with my divs in order to gain control of vertical alignment has cursed me with major psychological issues that I will never shake as long as this debate rages on.

The fact that people must frequently come up with complex and ugly workarounds to accomplish simple design goals (such as vertical alignment) strongly suggests that the rules are not nearly flexible enough. If the specs ARE sufficient, then why do high-profile sites (like SO) find it necessary to bend the rules using tables and other workarounds?

Share

answered [Aug 9, 2010 at 13:10](#)

community wiki
[DLH](#)

12

votes



I guess it's true that using the table element for layout has little to do with tabular data. So what?
Does my boss care? Do my users care?

Google and other automated systems **do** care, and they're just as important in many situations. Semantic code is easier for a non-intelligent system to parse and process.

Share

answered [Sep 17, 2008 at 13:23](#)

community wiki

[ceejayoz](#)

Yeah, for example for blogs the engine separate the comments from the blog post. Imagine the layout was styled with tables..
– [Omar Abid](#) Feb 8, 2010 at 10:57

2 -1: Google absolutely does not care in the slightest if you use tables for layout. – [Tom Anderson](#) Jan 12, 2011 at 15:38

@Tom Anderson Not because they have an issue with using tables for layout, but simply because non-table HTML tends to be more semantic. – [ceejayoz](#) Jan 12, 2011 at 18:42

8

votes



Having had to work with a website that involved 6 layers of nested tables generated by some application, and having had it generate invalid HTML, it was in fact a 3 hour job to rectify it breaking for a minor change.

This is of course the edge case, but table based design is unmaintainable. If you use css, you separate the style out so when fixing the HTML you have less to worry about breaking.

Also, try this with JavaScript. Move a single table cell from one place to another place in another table. Rather complicated to perform where div/span would just work copy-paste-wise.

"Does my boss care"

If I were your boss. You would care. ;) If you value your life.

Share

answered [Sep 17, 2008 at 13:26](#)

community wiki

[Kent Fredric](#)

Having had to work with a website that had an enormous soup of span and div tags and having witnessed the fragility of it when changing a single element would break the whole page (and divs used instead of heading tags)... – [inkredibl](#) Dec 22, 2010 at 10:07

Using Div's doesn't obviously make your code magically nicer. Much is to be said for appropriate semantic use of tags.
– [Kent Fredric](#) Dec 23, 2010 at 11:14

7 Layout flexibility

votes

Imagine you're making a page with a large number of



thumbnails.



DIVs:

If you put each thumbnail in a DIV, floated left, maybe 10 of them fit on a row. Make the window narrower, and BAM - it's 6 on a row, or 2, or however many fit.

TABLE:

You have to explicitly say how many cells are in a row. If the window is too narrow, the user has to scroll horizontally.

Maintainability

Same situation as above. Now you want to add three thumbnails to the third row.

DIVs:

Add them in. The layout will automatically adjust.

TABLE: Paste the new cells into the third row. **Oops!** Now there are too many items there. Cut some from that row and put them on the fourth row. Now there are too many items there. Cut some from that row... (etc)

(Of course, if you're generating the rows and cells with server-side scripting, this probably won't be an issue.)

Share

answered [Sep 17, 2008 at 15:09](#)

community wiki

[Nathan Long](#)

7

votes



I think that boat has sailed. If you look at the direction the industry has taken you will notice that CSS and Open Standards are the winners of that discussion. Which in turn



means for most html work, with the exception of forms, the designers will use divs instead of tables. I have a hard time with that because I am not a CSS guru but thats the way it is.

Share

answered [Sep 17, 2008 at 15:43](#)

community wiki
[Thomas Wagner](#)

5

votes



Also, don't forget, tables don't quite render well on mobile browsers. Sure, the iPhone has a kick-ass browser but everyone doesn't have an iPhone. Table rendering can be peanuts for modern browsers, but it's a bunch of watermelons for mobile browsers.

I have personally found that many people use too many `<div>` tags, but in moderation, it can be extremely clean and easy to read. You mention that folks have a harder time reading CSS than tables; in terms of 'code' that maybe true; but in terms of reading content (view > source) it is a heck of a lot easier to understand the structure with stylesheets than with tables.

Share



answered [Sep 17, 2008 at 13:30](#)

community wiki
[Swati](#)

Good point you have there; but IMHO the UI for mobile devices should be completely different taking into account the input limitations. – [Manrico Corazzi](#) Sep 17, 2008 at 15:55

True; which is why I've started using a different approach at times. In the MVC model, have my view pop-out just XML raw data i.e. content. Then begin with another MVC model where the xml raw data = model; view = html/css; controller = xsl. The XSL stylesheet purges unnecessary data for mobile. – [Swati](#) Sep 17, 2008 at 16:00

5
votes

Looks like you are just used to tables and that's it. Putting layout in a table limits you for just that layout. With CSS you can move bits around, take a look at <http://csszengarden.com/> And no, layout does not usually require a lot of nested divs.

With no tables for layout and proper semantics HTML is much cleaner, hence easier to read. Why should someone who cannot understand CSS try to read it? And if someone considers himself to be webdeveloper then the good grasp of CSS is a must.

SEO benefits come from the ability to have most important content higher up the page and having better content-to-markup ratio.

<http://www.hotdesign.com/seybold/>

Share

answered [Sep 17, 2008 at 13:31](#)

community wiki

[Rimantas](#)

5

votes



- 508 Compliance - the ability for a screenreader to make sense of your markup.
- Waiting for render - tables don't render in the browser until it gets to the end of the `</table>` element.

Share

answered [Sep 17, 2008 at 14:10](#)

community wiki

[Rick Glos](#)

I remember creating enormous tables years ago, back in the days of slower computers, and I remember when the code came in that made them render as you went. IE6? IE4? Can't remember, but it certainly did change. Of course, this is useful for tabulated data, but layout tables are styled to within an inch of their lives, so perhaps progressive rendering would be less useful as the table jumps around to accommodate the newly loaded content. – [ijw](#) Nov 29, 2010 at 14:03

5

votes



The whole idea around semantic markup is the separation of markup and presentation, which includes layout.



Div's aren't replacing tables, they have their own use in separating content into blocks of related content (,). When you don't have the skills and are relying on tables, you'll often have to separate your content in to cells in order to get the desired layout, but you wont need to touch the markup to achieve presentation when using semantic markup. This is really important when the markup is being generated rather than static pages.

Developers need to stop providing markup that implies layout so that those of us who do have the skills to present content can get on with our jobs, and developers don't have to come back to their code to make changes when presentation needs change.

Share

answered [Sep 17, 2008 at 14:52](#)

community wiki
[Steve Perks](#)

4

votes



This isn't really about whether 'divs are better than tables for layout'. Someone who understands CSS can duplicate any design using 'layout tables' pretty straightforwardly. The real win is using HTML elements for what they are there for. The reason you would not use tables for non-tablular data is the same reason you don't store integers as character strings - technology works much more easily when you use it for the purpose for which it is desgined. If it was ever

necessary to use tables for layout (because of browser shortcomings in the early 1990s) it certainly isn't now.

Share

answered [Sep 17, 2008 at 23:03](#)

community wiki
[domgblackwell](#)

3

votes



Tools that use table layouts can become extraordinarily heavy due to the amount of code required to create the layout. SAP's Netweaver Portal by default uses TABLE to layout their pages.

The production SAP portal at my current gig has a home page whose HTML weighs over 60K and goes seven tables deep, three times within the page. Add in the Javascript, the misuse of 16 iframes with similar table issues inside of them, overly heavy CSS etc, and the page weighs over 5MB.

Taking the time to lower the page weight so you can use your bandwidth to do engaging activities with users is worth the effort.

Share

edited [Sep 17, 2008 at 15:22](#)

community wiki
[2 revs](#)
[Mike Cornell](#)

3

votes



It's worth figuring out CSS and divs so the central content column loads and renders before the sidebar in a page layout. But if you are struggling to use floating divs to vertically align a logo with some sponsorship text, just use the table and move on with life. The Zen garden religion just doesn't give much bang for the buck.

The idea of separating content from presentation is to partition the application so different kinds of work affect different blocks of code. This is actually about change management. But coding standards can only examine the present state of code in a superficial manner.

The change log for an application that depends on coding standards to "separate content from presentation" will show a pattern of parallel changes across vertical silos. If a change to "content" is always accompanied by a change to "presentation", how successful is the partitioning?

If you really want to partition your code productively, use Subversion and review your change logs. Then use the simplest coding techniques -- divs, tables, JavaScript, includes, functions, objects, continuations, whatever -- to structure the application so that the changes fit in a simple and comfortable manner.

Share

edited May 1, 2010 at 21:51

community wiki
2 revs, 2 users 71%
Patrick May

+1 for the first two sentences. – [Sean McMillan](#) Jul 27, 2011 at 16:29

3
votes



Because it's HELL to maintain a site that uses tables, and takes a LOT longer to code. If you're scared of floating divs, go take a course in them. They're not difficult to understand and they're approximately 100 times more efficient and a million times less a pain in the ass (unless you don't understand them -- but hey, welcome to the world of computers).

Anyone considering doing their layout with a table better not expect me to maintain it. It's the most ass-backwards way to render a website. Thank god we have a much better alternative now. I would NEVER go back.

It's scary that some folks might not be aware of the time and energy benefits from creating a site using modern tools.

Share

answered [Nov 29, 2010 at 18:45](#)

community wiki
[Chuck Le Butt](#)

3

votes



Tables are not *in general* easier or more maintainable than CSS. However, there are a few *specific* layout-problems where tables are indeed the simplest and most flexible solution.

CSS is clearly preferable in cases where presentational markup and CSS support the same kind of design, no one in their right mind would argue that `font`-tags are better than specifying typography in CSS, since CSS gives you the same power than `font`-tags, but in a much cleaner way.

The issue with tables, however, is basically that the table-layout model in CSS is not supported in Microsoft Internet Explorer. Tables and CSS are therefore *not* equivalent in power. The missing part is the **grid-like behavior** of tables, where the edges of cells align both vertically and horizontally, while cells still expand to contain their content. This behavior is not easy to achieve in pure CSS without hardcoding some dimensions, which makes the design rigid and brittle (as long as we have to support Internet Explorer - in other browsers this is easily achieved by using `display:table-cell`).

So it's not really a question of whether tables or CSS is preferable, but it is a question of recognizing the specific cases where use of tables may make the layout more flexible.

The most important reason for *not* using tables is accessibility. The Web Content Accessibility Guidelines

<http://www.w3.org/TR/WCAG10/> advice against using tables for layout. If you are concerned about accessibility (and in some cases you may be legally obliged to), you should use CSS even if tables are simpler. Note that you can *always* create the same layout with CSS as with tables, it might just require more work.

Share

edited Dec 13, 2010 at 17:21

community wiki
5 revs, 2 users 79%
JacquesB

3

votes



I was surprised to see some issues were not already covered, so here are my 2 cents, in addition to all the very valid points made earlier:

.1. **CSS & SEO:**

a) CSS used to have a very significant impact on SEO by allowing to position the content in the page wherever you want. A few years ago, Search Engines were giving a significant emphasis to "on-page" factors. Something at the top of the page was deemed more relevant to the page than something located at the bottom. "Top of the page" for a spider meant "at the beginning of the code". Using CSS, you could organize your keyword-rich content at the beginning of the code, and still position it wherever you liked in the page. This is still somewhat relevant, but on page factors are less and less important for page ranking.

b) When the layout is moved over to CSS, the HTML page is lighter and therefore loads faster for a search engine spider. (spiders don't bother downloading external css files). Fast loading pages is an important ranking consideration for several search engines, including Google

c) SEO work often requires testing and changing things, which is much more convenient with a CSS based layout

.2. Generated content:

A table is considerably easier to generate programmically than the equivalent CSS layout.

```
foreach ($comment as $key=>$value)
{
    echo "<tr><td>$key</td><td>$value</td></tr>";
}
```

Generating a table is simple and safe. It is self-contained and integrates well within any template. To do the same with CSS is considerably harder and may be of no benefit at all: hard to edit the CSS stylesheet on the flight, and adding the style inline is no different from using a table (content is not separated from layout).

Further, when a table is generated, the content (in variables) is already separated from the layout (in code), making it as easy to modify.

This is one reason why some very well designed websites (SO for instance) still use table layouts.

Of course, if the results need to be acted upon through JavaScript, divs are worth the trouble.

.3. Quick conversion testing

When figuring out what works for a specific audience, it is useful to be able to change the layout in various ways to figure out what gets the best results. A CSS based layout makes things considerably easier

.4. Different solutions for different problems

Layout tables are usually dissed because "everybody knows divs & CSS" are the way to go.

However the fact remains that tables are faster to create, easier to understand and are more robust than most CSS layouts. (Yes, CSS can be as robust, but a quick look through the net on different browsers and screen resolutions shows it's not often the case)

There are a lot of downsides to tables, including maintenance, lack of flexibility... but let's not throw the baby with the bath water. There are plenty of professional uses for a solution which is both quick and reliable.

Some time ago, I had to rewrite a clean and simple CSS layout using tables because a significant portion of the users would be using an older version of IE with really bad support for CSS

I, for one, am sick and tired of the knee-jerk reaction "Oh noes! Tables for layout!"

As for the "it wasn't intended for that purpose and therefore you shouldn't use it this way" crowd, isn't that hypocrisy? What do you think of all the CSS tricks you have to use to get the darn thing working in most browsers? Were they meant for that purpose?

Share

answered [Apr 11, 2011 at 11:59](#)

community wiki
[Sylver](#)

1

2

3

Next