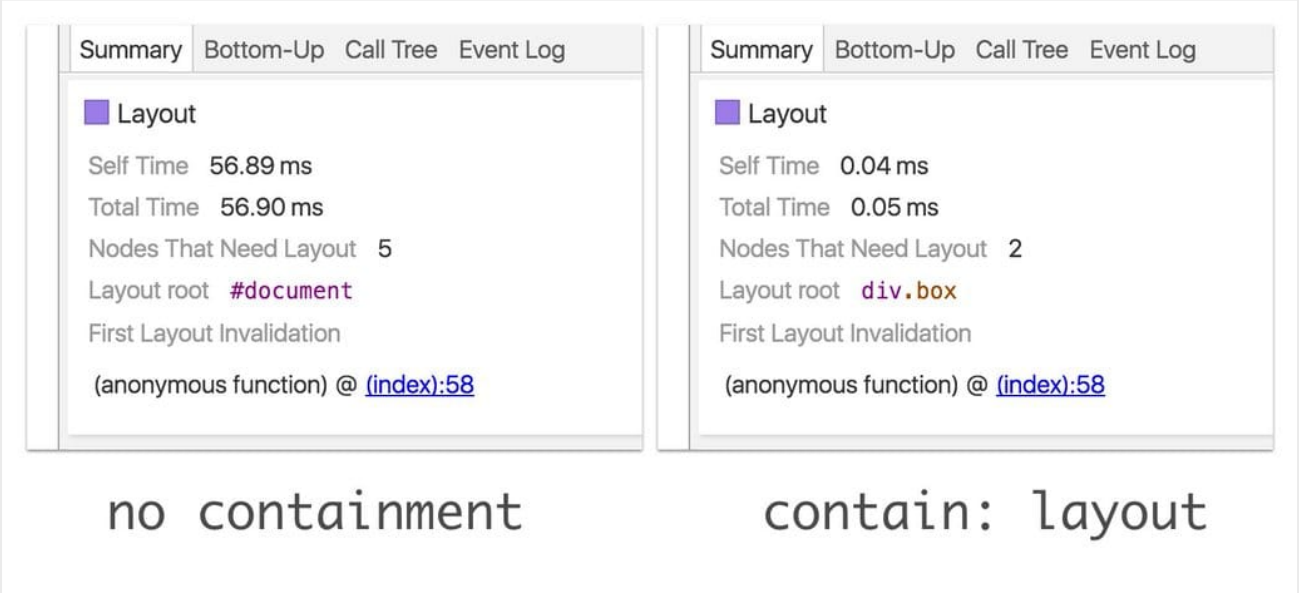
CSS Containment in Chrome 52

By Paul Lewis
Paul is a Design and Perf Advocate

TL;DR

The new CSS Containment property lets developers limit the scope of the browser's styles, layout and paint work.



It has a few values, making its syntax this:

```
contain: none | strict | content | [ size || layout || style || paint ]
```

It's in Chrome 52+ and Opera 40+ (and it has public support from Firefox), so give it a whirl and let us know how you go!

The contain property

When making a web app, or even a complex site, a key performance challenge is limiting the effects of styles, layout and paint. Oftentimes the *entirety* of the DOM is considered "in scope" for computation work, which can mean that attempting a self-contained "view" in a

web app can prove tricky: changes in one part of the DOM can affect other parts, and there's no way to tell the browser what should be in or out of scope.

For example, let's say part of your DOM looks like this:

```
<section class="view">
  Home
</section>

<section class="view">
  About
</section>

<section class="view">
  Contact
</section>
```

And you append a new element to one view, which will trigger styles, layout and paint:

```
<section class="view">
  Home
</section>

<section class="view">
  About
  <div class="newly-added-element">Check me out!</div>
</section>

<section class="view">
  Contact
</section>
```

In this case, however, the *whole DOM* is effectively in scope, meaning that style, layout, and paint calculations will have to consider *all the elements* irrespective of whether or not they were changed. The bigger the DOM, the more computation work that involves, meaning that you could well make your app unresponsive to user input.

The good news is that modern browsers are getting really smart about limiting the scope of styles, layout, and paint work automatically, meaning that things are getting faster without you having to do anything.

But the *even better* news is that there's a new CSS property that hands scope controls over to developers: **Containment**.

The contain property

CSS Containment is a new property, with the keyword contain, which supports four values:

- `layout`
- `paint`
- `size`
- `style`

Each of these values allows you to limit how much rendering work the browser needs to do. Let's take a look at each in a little more detail.

Layout (contain: layout)

> This value turns on layout containment for the element. This ensures that the containing element is totally opaque for layout purposes; nothing outside can affect its internal layout, and vice versa. Containment spec

Layout containment is probably *the* biggest benefit of containment, along with `contain: paint`.

Layout is normally document-scoped, making it scale proportionally to the size of your DOM, so if you change an element's `left` property (as just one example), every single element in the DOM might need to be checked.

Enabling containment here can potentially reduce the number of elements to just a handful, rather than the whole document, saving the browser a ton of unnecessary work and significantly improving performance.

Paint (contain: paint)

> This value turns on paint containment for the element. This ensures that the descendants of the containing element don't display outside its bounds, so if an element is off-screen or otherwise not visible, its descendants are also guaranteed to be not visible.
> Containment spec

Scoping paint is another incredibly useful benefit of containment. Paint containment essentially clips the element in question, but it also has a few other side effects:

- **It acts as a containing block for absolutely positioned and fixed position elements.** This means any children are positioned based on the element with `contain: paint` not any other parent element like – say – the document.
- **It becomes a stacking context.** This means that things like `z-index` will have an effect on the element, and children will be stacked according to the new context.
- **It becomes a new formatting context.** This means that if you have, for example, a block level element with paint containment, it will be treated as a new, *independent* layout environment. This means that layout outside of the element won't typically affect the containing element's children.

Size (contain: size)

> The value turns on size containment for the element. This ensures that the containing element can be laid out without needing to examine its descendants. Containment spec

What `contain: size` means is that the element's children *do not affect the parent's size*, and that its inferred or declared dimensions will be the ones used. Consequently if you were to set `contain: size` but didn't specify dimensions for the element (either directly or via flex properties), it would be rendered at 0px by 0px!

Size containment is really a belt-and-braces measure to ensure you don't rely on child elements for sizing, but by itself it doesn't offer much performance benefit.

Style (contain: style)

> This value turns on style containment for the element. This ensures that, for properties which can have effects on more than just an element and its descendants, those effects don't escape the containing element. Containment spec

It can be hard to predict what the effects on the DOM tree of changing an element's styles will be back up the tree. One example of this is in something like CSS counters, where changing a counter in a child can affect counter values of the same name used elsewhere in the document. With `contain: style` set, style changes won't get propagated back up past the containing element.

To be super clear, what `contain: style` *doesn't* provide is scoped styling as you'd get from Shadow DOM; containment here is purely about limiting the parts of the tree that are under consideration when styles are mutated, *not* when they are declared.

Strict and content containment

You can also combine keywords, such as `contain: layout paint`, which will apply only those behaviors to an element. But contain also supports two additional values:

- `contain: strict` means the same as `contain: layout style paint size`
- `contain: content` means the same as `contain: layout style paint`

Using strict containment is great when you know the size of the element ahead of time (or wish to reserve its dimensions), but bear in mind that if you declare strict containment *without* dimensions, because of the implied size containment, the element may be rendered as a 0px by 0px box.

Content containment, on the other hand, offers significant scope improvements, but does not require you to know or specify the dimensions of the element ahead of time.

Of the two, `contain: content` is the one you should look to use by default. You should treat strict containment as something of a more an escape hatch when `contain: content` isn't strong enough for your needs.

Let us know how you get on

Containment is a great way to start indicating to the browser what you intend to be kept isolated within your page. Give it a try in Chrome 52+ and let us know how you get on!

Useful links:

- CSS Containment Spec
- Containment on Chrome Status

Last updated July 2, 2018.