



Using the viewport meta tag to control layout on mobile browsers

[See Mobile](#)

English ▼

[Jump to section](#) ▼

Background

The browser's [viewport](#) is the area of the window in which web content can be seen. This is often not the same size as the rendered page, in which case the browser provides scrollbars for the user to scroll around and access all the content.

Narrow screen devices (e.g. mobiles) render pages in a virtual window or viewport, which is usually wider than the screen, and then shrink the rendered result down so it can all be seen at once. Users can then pan and zoom to see different areas of the page. For example, if a mobile screen has a width of 640px, pages might be rendered with a virtual viewport of 980px, and then it will be shrunk down to fit into the 640px space.

This is done because many pages are not mobile optimized, and break (or at least look bad) when rendered at a small viewport width. This virtual viewport is a way to make non-mobile-optimized sites in general look better on narrow screen devices.

Enter viewport meta tag

However, this mechanism is not so good for pages that are optimized for narrow screens using [media queries](#) — if the virtual viewport is 980px for example, media queries that kick in at 640px or 480px or less will never be used, limiting the effectiveness of such responsive design techniques.

To mitigate this problem, Apple introduced the "viewport meta tag" in Safari iOS to let web developers control the viewport's size and scale. Many other mobile browsers now support this tag, although it is not part of any web standard. Apple's [documentation](#) does a good job explaining how web developers can use this tag, but we had to do some detective work to figure out exactly how to implement it in Fennec. For example, Safari's documentation says the content is a "comma-delimited list," but existing browsers and web pages use any mix of commas, semicolons, and spaces as separators.

Learn more about viewports in different mobile browsers in [A Tale of Two Viewports](#) at quirksmode.org.

Viewport basics

A typical mobile-optimized site contains something like the following:

```
<meta name="viewport" content="width=device-width, initial-scale=1">
```

The `width` property controls the size of the viewport. It can be set to a specific number of pixels like `width=600` or to the special value `device-width`, which is the width of the screen in CSS pixels at a scale of 100%. (There are corresponding `height` and `device-height` values, which may be useful for pages with elements that change size or position based on the viewport height.)

The `initial-scale` property controls the zoom level when the page is first loaded. The `maximum-scale`, `minimum-scale`, and `user-scalable` properties control how users are allowed to zoom the page in or out.

❗ Usage of `user-scalable=no` can cause accessibility issues to users with visual impairments such as low vision.

A pixel is not a pixel

In recent years, screen resolutions have risen to the size that individual pixels are hard to distinguish with the human eye. For example, recent smartphones generally have a 5-inch screens with resolutions upwards of 1920—1080 pixels (~400 dpi). Because of this, many browsers can display their pages in a smaller physical size by translating multiple hardware pixels for each CSS "pixel". Initially this caused usability and readability problems on many touch-optimized web sites. Peter-Paul Koch wrote about this problem in [A pixel is not a pixel](#).

On high dpi screens, pages with `initial-scale=1` will effectively be zoomed by browsers. Their text will be smooth and crisp, but their bitmap images will probably not take advantage of the full screen resolution. To get sharper images on these screens, web developers may want to design images – or whole layouts – at a higher scale than their final size and then scale them down using CSS or viewport properties. This is consistent with the [CSS 2.1 specification](#), which says:

If the pixel density of the output device is very different from that of a typical computer display, the user agent should rescale pixel values. It is recommended that the pixel unit refer to the whole number of device pixels that best approximates the reference pixel. It is recommended that the reference pixel be the visual angle of one pixel on a device with a pixel density of 96dpi and a distance from the reader of an arm's length.

For web developers, this means that the size of a page is much smaller than the actual pixel count and browsers may size their layouts and images accordingly. But remember that not all

mobile devices are the same width; you should make sure that your pages work well in a large variation of screen sizes and orientations.

The default pixel ratio depends on the display density. On a display with density less than 200dpi, the ratio is 1.0. On displays with density between 200 and 300dpi, the ratio is 1.5. For displays with density over 300dpi, the ratio is the integer floor($density/150dpi$). Note that the default ratio is true only when the viewport scale equals 1. Otherwise, the relationship between CSS pixels and device pixels depends on the current zoom level.

Viewport width and screen width

Sites can set their viewport to a specific size. For example, the definition `"width=320, initial-scale=1"` can be used to fit precisely onto a small phone display in portrait mode. This can cause [problems](#) when the browser doesn't render a page at a larger size. To fix this, browsers will expand the viewport width if necessary to fill the screen at the requested scale. This is especially useful on large-screen devices like the iPad. (Allen Pike's [Choosing a viewport for iPad sites](#) has a good explanation for web developers.)

For pages that set an initial or maximum scale, this means the `width` property actually translates into a *minimum* viewport width. For example, if your layout needs at least 500 pixels of width then you can use the following markup. When the screen is more than 500 pixels wide, the browser will expand the viewport (rather than zoom in) to fit the screen:

```
<meta name="viewport" content="width=500, initial-scale=1">
```

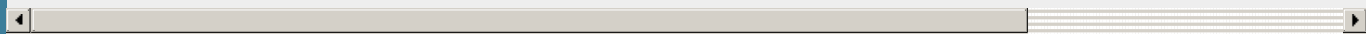
Other [attributes](#) that are available are `minimum-scale`, `maximum-scale`, and `user-scalable`. These properties affect the initial scale and width, as well as limiting changes in zoom level.

Not all mobile browsers handle orientation changes in the same way. For example, Mobile Safari often just zooms the page when changing from portrait to landscape, instead of laying out the page as it would if originally loaded in landscape. If web developers want their scale settings to remain consistent when switching orientations on the iPhone, they must add a `maximum-scale` value to prevent this zooming, which has the sometimes-unwanted side effect of preventing users from zooming in:

```
<meta name="viewport" content="initial-scale=1, maximum-scale=1">
```

Suppress the small zoom applied by many smartphones by setting the initial scale and minimum-scale values to 0.86. The result is horizontal scroll is suppressed in any orientation and the user can zoom in if they want to.

```
<meta name="viewport" content="width=device-width, initial-scale=0.86, maximum-scale=3.0">
```



Common viewport sizes for mobile and tablet devices

If you want to know what mobile and tablet devices have which viewport widths, there is a comprehensive list of [mobile and tablet viewport sizes here](#). This gives information such as viewport width on portrait and landscape orientation as well as physical screen size, operating system and the pixel density of the device.

Specifications

Specification	Status	Comment
CSS Device Adaptation The definition of '<meta name="viewport">' in that specification.	<div><div>WD</div>Working Draft</div>	Non-normatively describes the Viewport META element

There is clearly demand for the viewport meta tag, since it is supported by most popular mobile browsers and used by thousands of web sites. It would be good to have a true standard for web pages to control viewport properties. As the standardization process proceeds, we at Mozilla will work to keep up to date with any changes.

🕒 Last modified: Jul 20, 2020, by [MDN contributors](#)

×

Learn the best of web development

Get the latest and greatest from MDN delivered straight to your inbox.

☐ I'm okay with Mozilla handling my info as explained in this [Privacy Policy](#).

[Sign up now](#)



Web Technologies

[Learn Web Development](#)

[About MDN](#)

[Feedback](#)

[About](#)

[MDN Web Docs Store](#)

[Contact Us](#)

[Firefox](#)

[MDN](#)



[Mozilla](#)



© 2005-2020 Mozilla and individual contributors. Content is available under these licenses.

[Terms](#)

[Privacy](#)

[Cookies](#)