CSSWG F2F
Gamut Mapping

ccameron-chromium, 2024-02-14

The issue

The bug: github.com/w3c/csswg-drafts/issues/9449

This codepen with `oklch(90% 10% 0deg)` and `oklch(90% 90% 0deg)`.

- “The promise of `(ok)lch` is supposedly that lightness should be consistent across different hues and chromas. Clearly that's not currently true.”
- “From an authoring perspective it's entirely unusable, and it breaks the fundamental promise of the format: providing perceptually-uniform lightness”
- “This is the format that authors were most excited about, and it doesn't do what we told them it does.”

The bug: github.com/w3c/csswg-drafts/issues/9449

This codepen with `oklch(90% 10% 0deg)` and `oklch(90% 90% 0deg)`.

- “The promise of `(ok)lch` is supposedly that lightness should be consistent across different hues and chromas. Clearly that's not currently true.”

- “From an authoring perspective it's entirely unusable, and it breaks the fundamental promise of the format: providing perceptually-uniform lightness”

- “This is the format that authors were most excited about, and it doesn't do what we told them it does.”

What `oklch` ***DOES*** guarantee

Suppose we have in-gamut colors `oklab(L0,a0,b0)` and `oklab(L1,a1,b1)`

- The line between them is perceptually uniform
- If `L0==L1`, the line has constant lightness
- If `(a0,b0)` and `(a1,b1)` are same length, the line has constant saturation
- If `(a0,b0)` and `(a1,b1)` are same angle, the line has constant hue

This is a good space to do interpolation in.

What `oklch` ***DOES NOT*** guarantee

Suppose `L` is in `[0%,100%]`, `c` is in `[0%,100%]` and `h` is in `[0deg,360deg]`

There is no guarantee that `oklch(L,c,h)` is in any particular gamut or even represents a physically possible color.

This is a **DANGEROUS** space to specify or manipulate colors in.

`oklab` and `oklch` are dangerous spaces to specify or manipulate color parameters

(their definition needs to be changed)

The codepen from the bug

Draws a gradient between

`oklch(90% 10% 0deg)` and

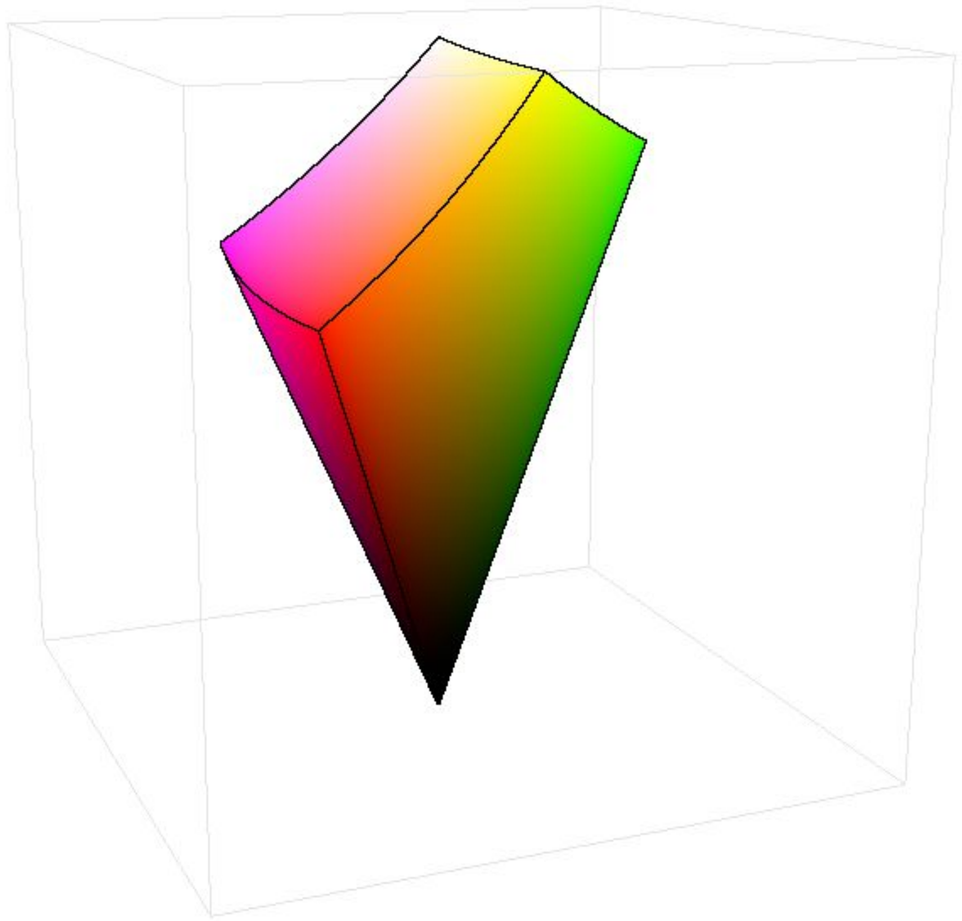
`oklch(90% 90% 0deg)`

The codepen from the bug

Draws a gradient between

`oklch(90% 10% 0deg)` and

`oklch(90% 90% 0deg)`

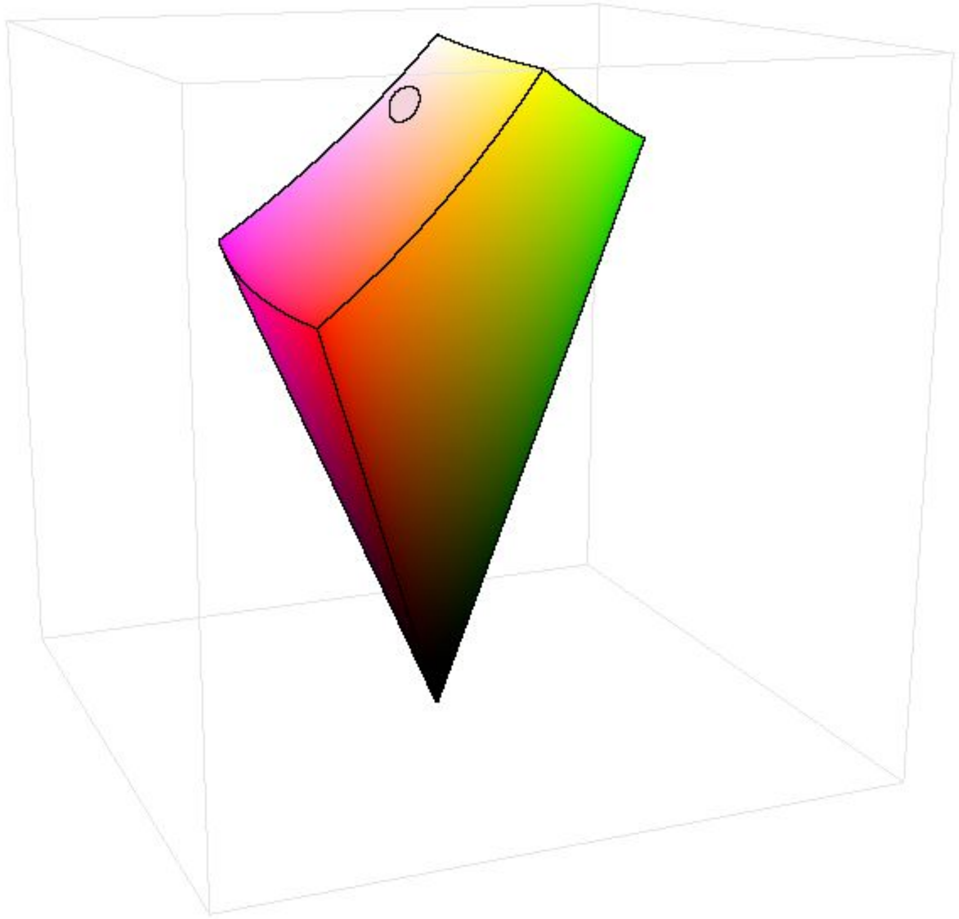


The codepen from the bug

Draws a gradient between

`oklch(90% 10% 0deg)` and

`oklch(90% 90% 0deg)`

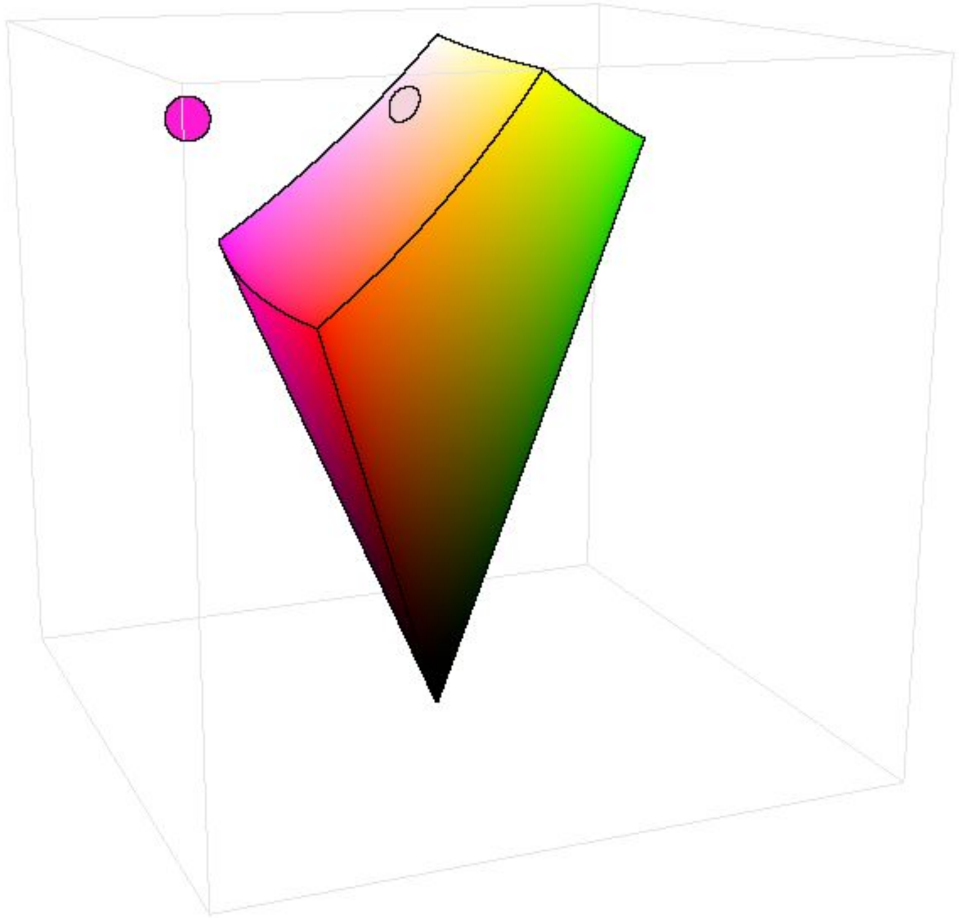


The codepen from the bug

Draws a gradient between

`oklch(90% 10% 0deg)` and

`oklch(90% 90% 0deg)`

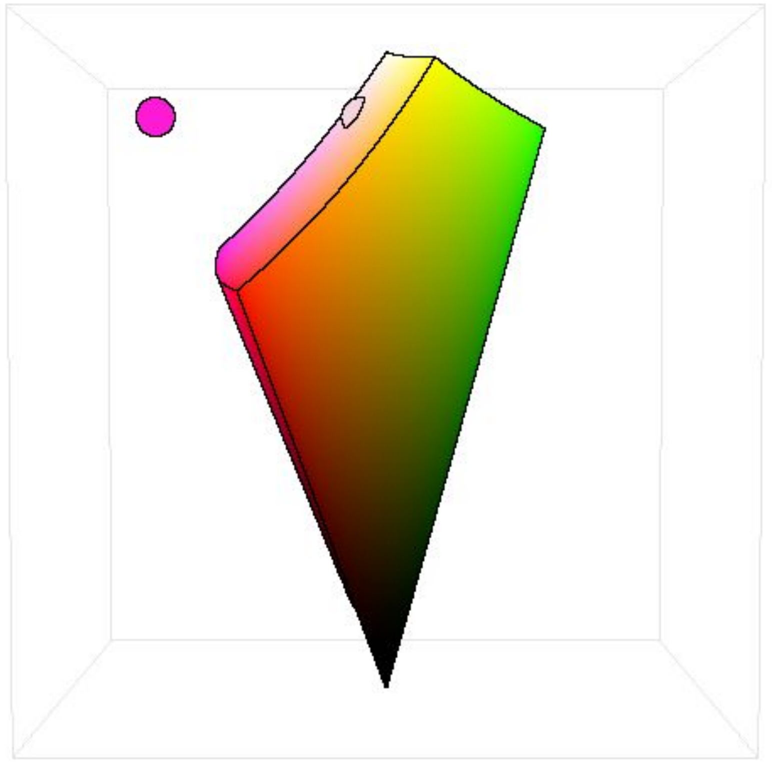


The codepen from the bug

Draws a gradient between

`oklch(90% 10% 0deg)` and

`oklch(90% 90% 0deg)`



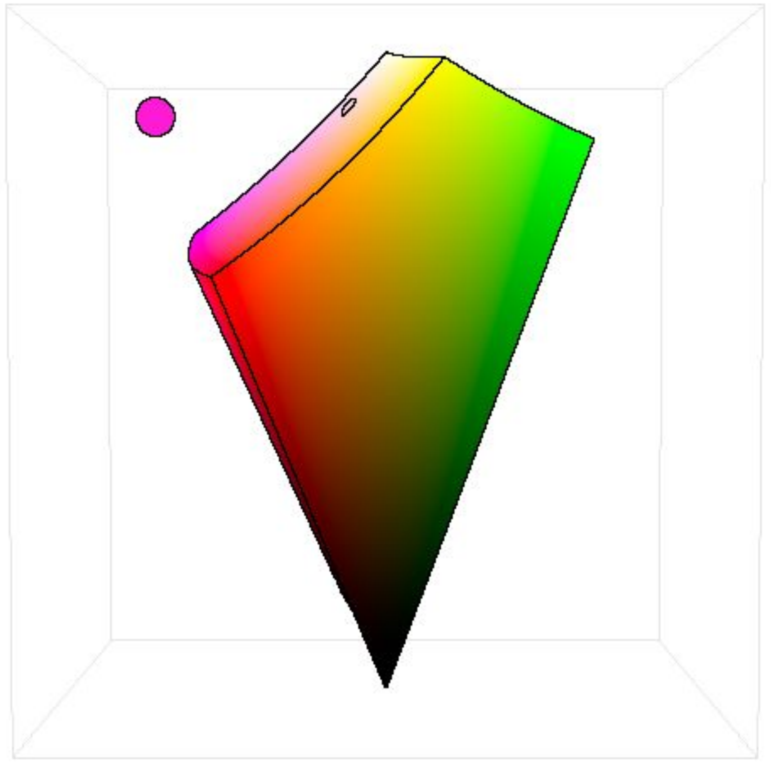
The codepen from the bug

Draws a gradient between

`oklch(90% 10% 0deg)` and

`oklch(90% 90% 0deg)`

(with P3 gamut)



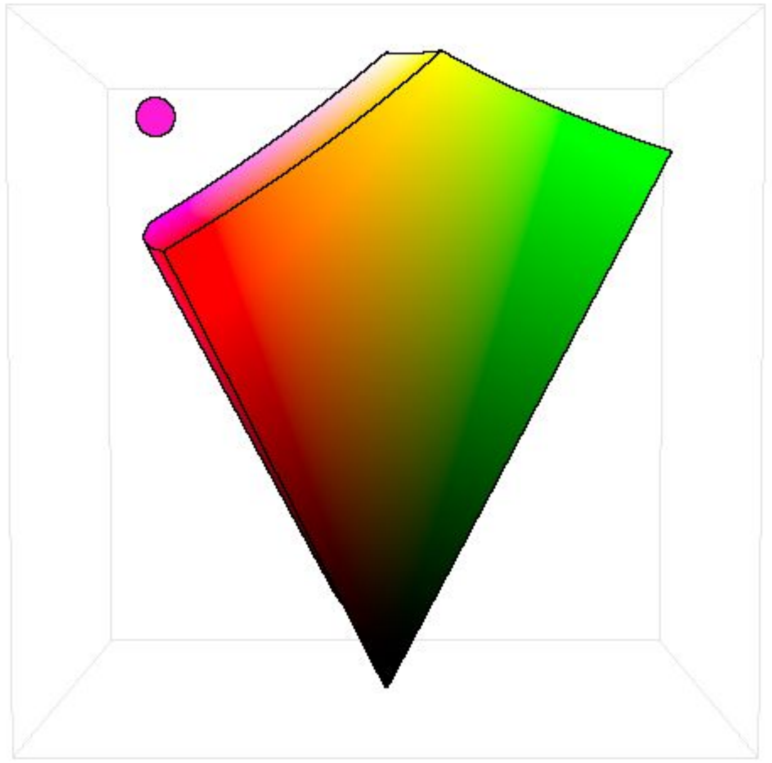
The codepen from the bug

Draws a gradient between

`oklch(90% 10% 0deg)` and

`oklch(90% 90% 0deg)`

(with Rec2020 gamut)

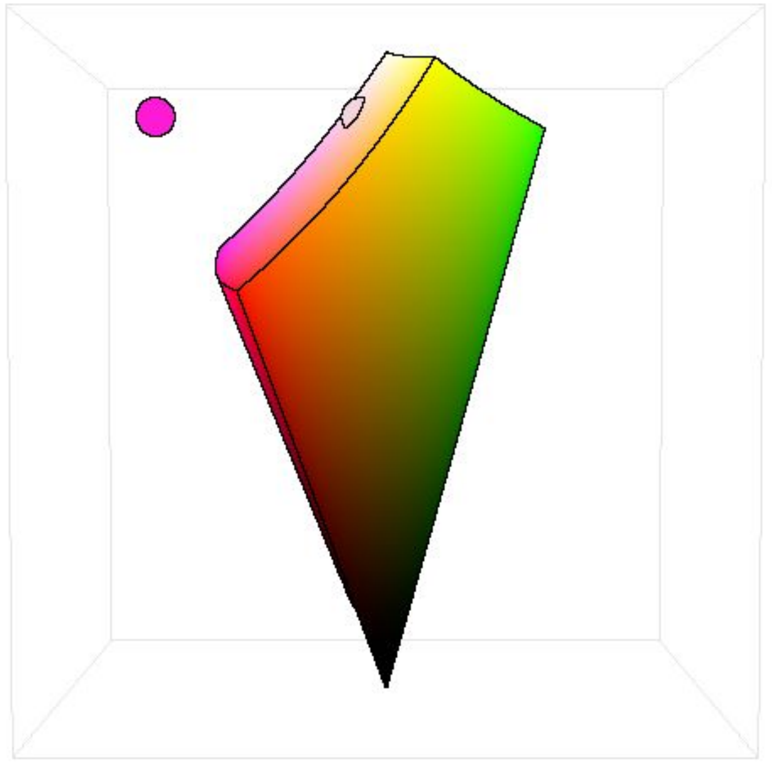


The codepen from the bug

Draws a gradient between

`oklch(90% 10% 0deg)` and

`oklch(90% 90% 0deg)`



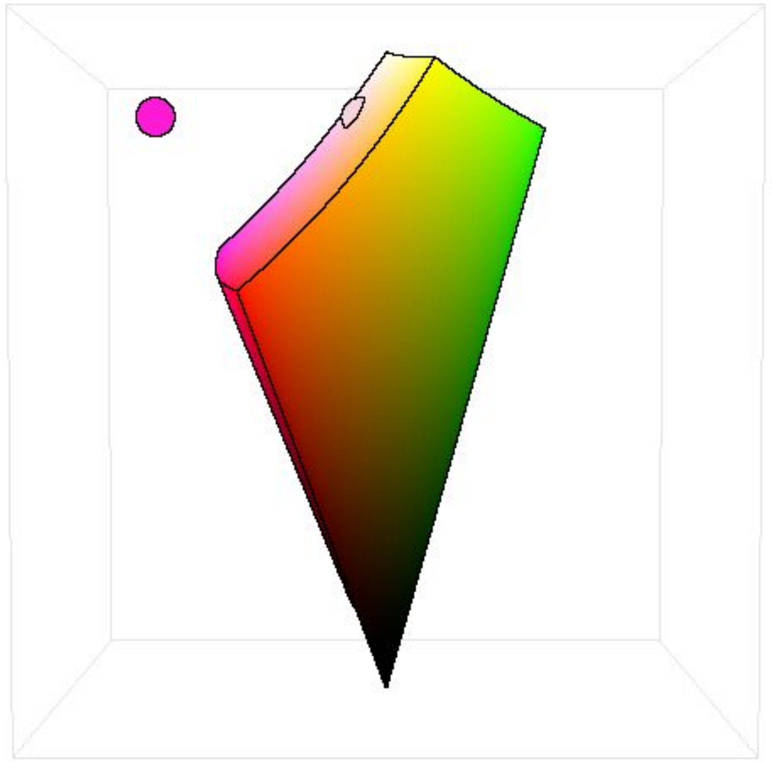
The codepen from the bug

Draws a gradient between

`oklch(90% 10% 0deg)` and

`oklch(90% 90% 0deg)`

NOT a safe parameter space for specifying values or manipulating values



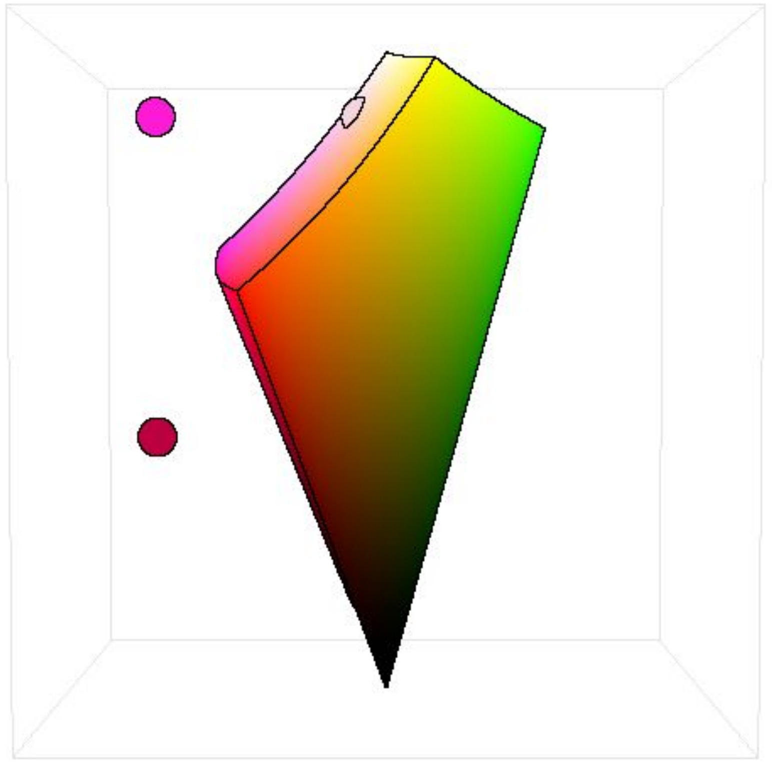
The codepen from the bug

Draws a gradient between

`oklch(90% 10% 0deg)` and

`oklch(90% 90% 0deg)` and

`oklch(10% 90% 0deg)`



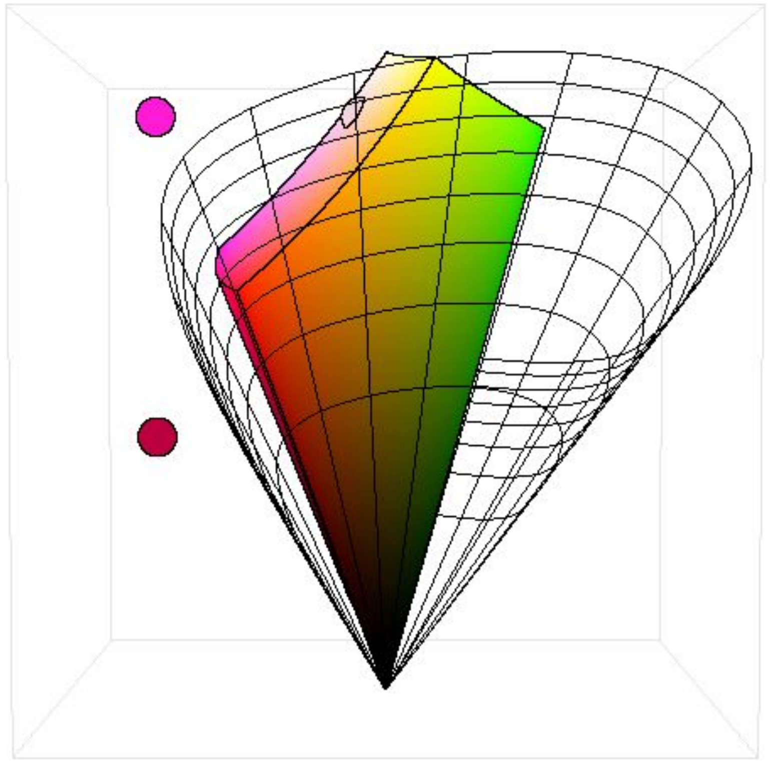
The codepen from the bug

Draws a gradient between

`oklch(90% 10% 0deg)` and

`oklch(90% 90% 0deg)` and

`oklch(10% 90% 0deg)`



What should `oklch(90% 90% 0deg)` look like?

(we need to limit what authors specify or everyone will suffer)

What should `oklch(90% 90% 0deg)` be?

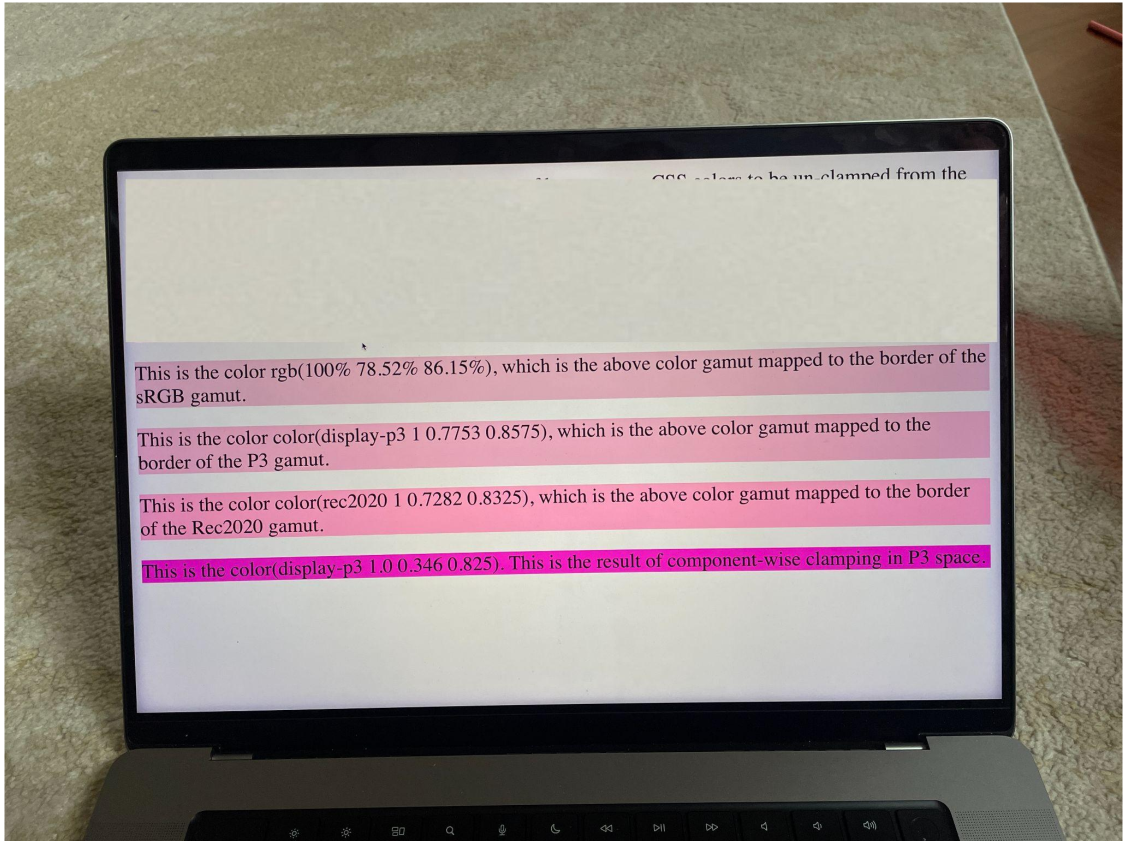
It is mathematically equivalent to `color(srgb 1.53, 0.10, 0.83)`.

CSS gamut mapping to sRGB gives us
`rgb(100% 78.52% 86.15%)`

Clamping to sRGB gives us
`rgb(100% 10.37% 83.26%)`

But what did we actually specify?

What should `oklch(90% 90% 0deg)` be?



CSS colors to be un-clamped from the

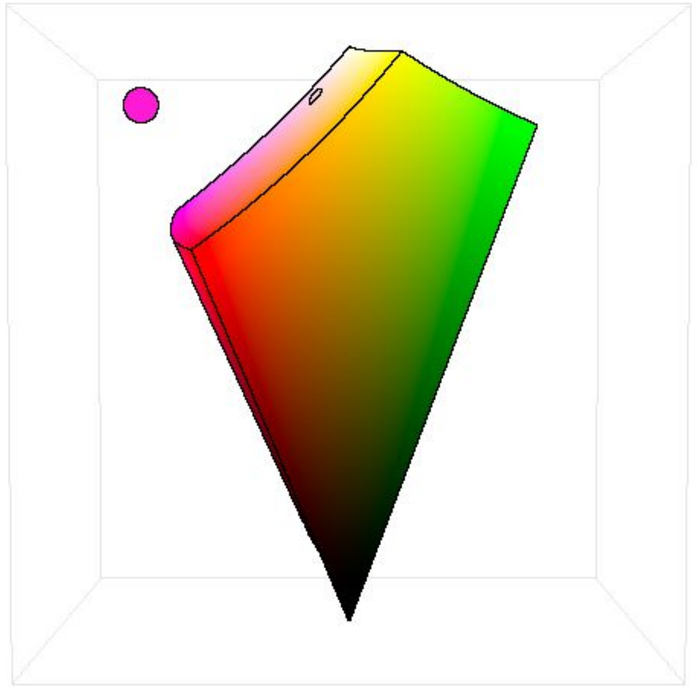
This is the color rgb(100% 78.52% 86.15%), which is the above color gamut mapped to the border of the sRGB gamut.

This is the color color(display-p3 1 0.7753 0.8575), which is the above color gamut mapped to the border of the P3 gamut.

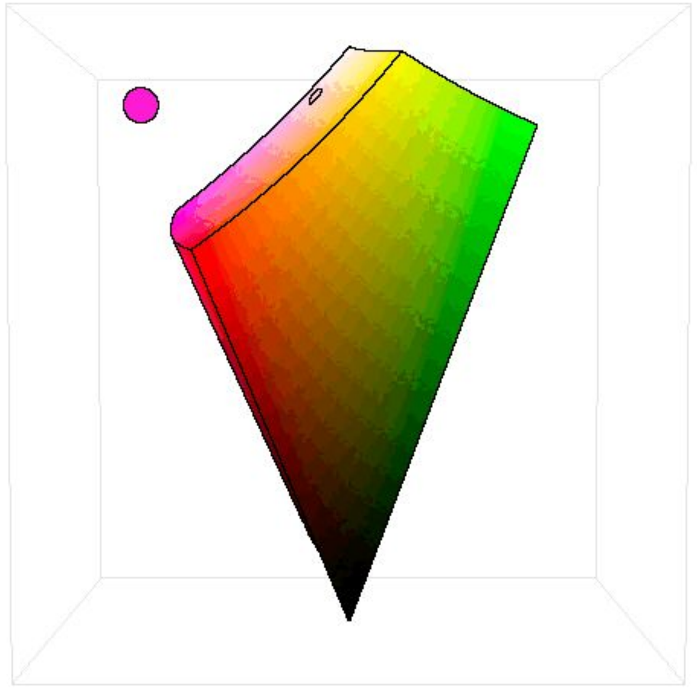
This is the color color(rec2020 1 0.7282 0.8325), which is the above color gamut mapped to the border of the Rec2020 gamut.

This is the color(display-p3 1.0 0.346 0.825). This is the result of component-wise clamping in P3 space.

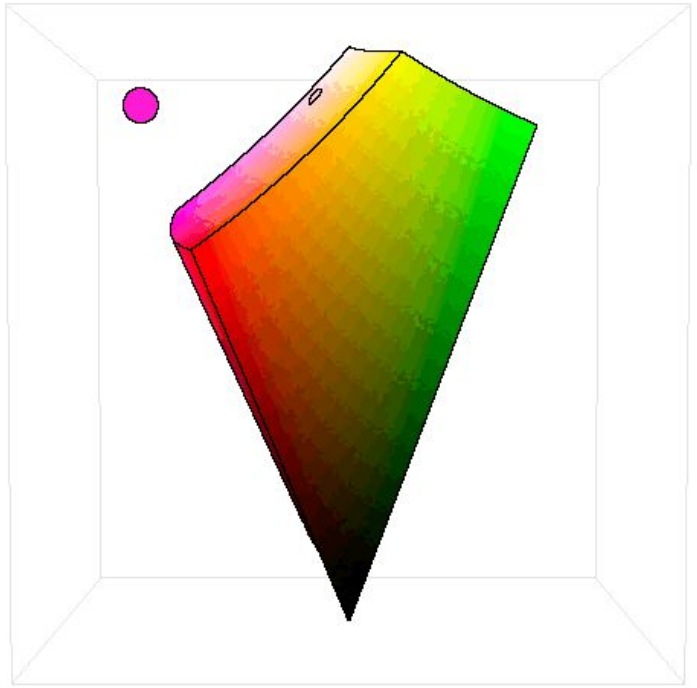
What should `oklch(90% 90% 0deg)` be?



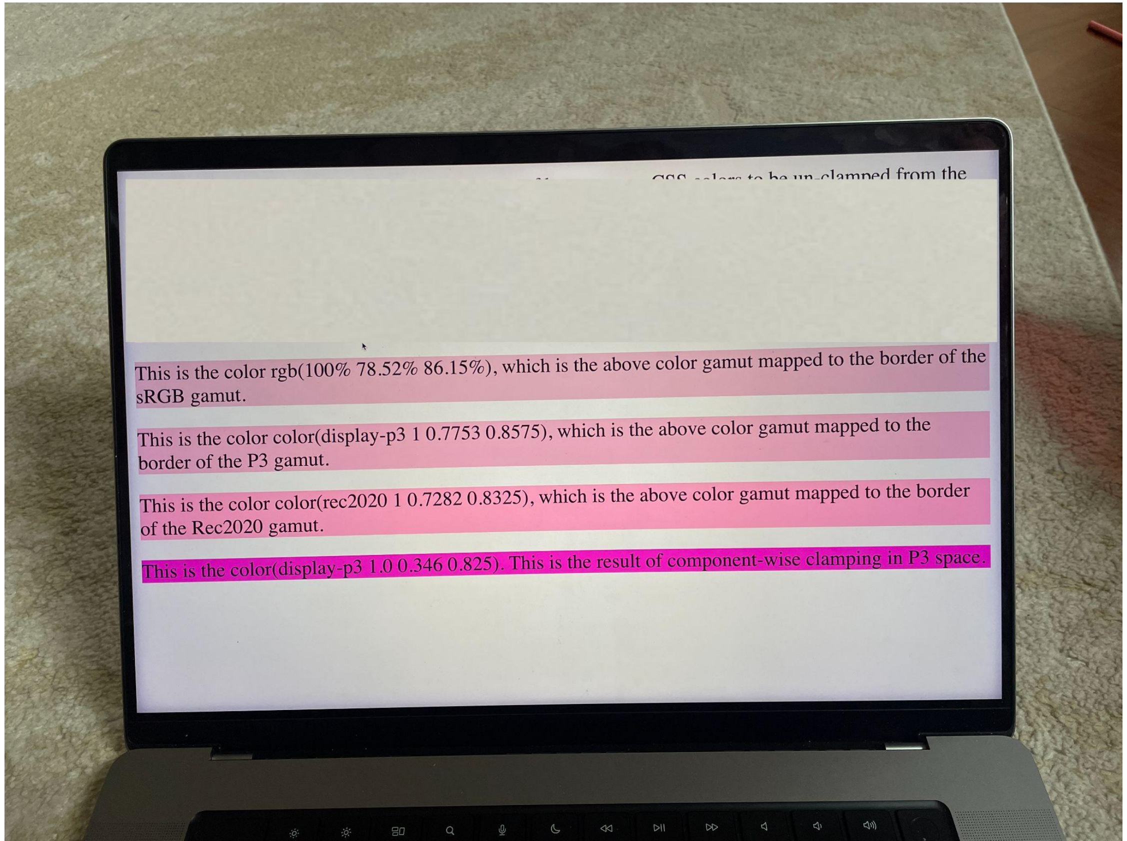
What should `oklch(90% 90% 0deg)` be?



What should `oklch(90% 90% 0deg)` be?



What should `oklch(90% 90% 0deg)` be?



CSS colors to be un-clamped from the

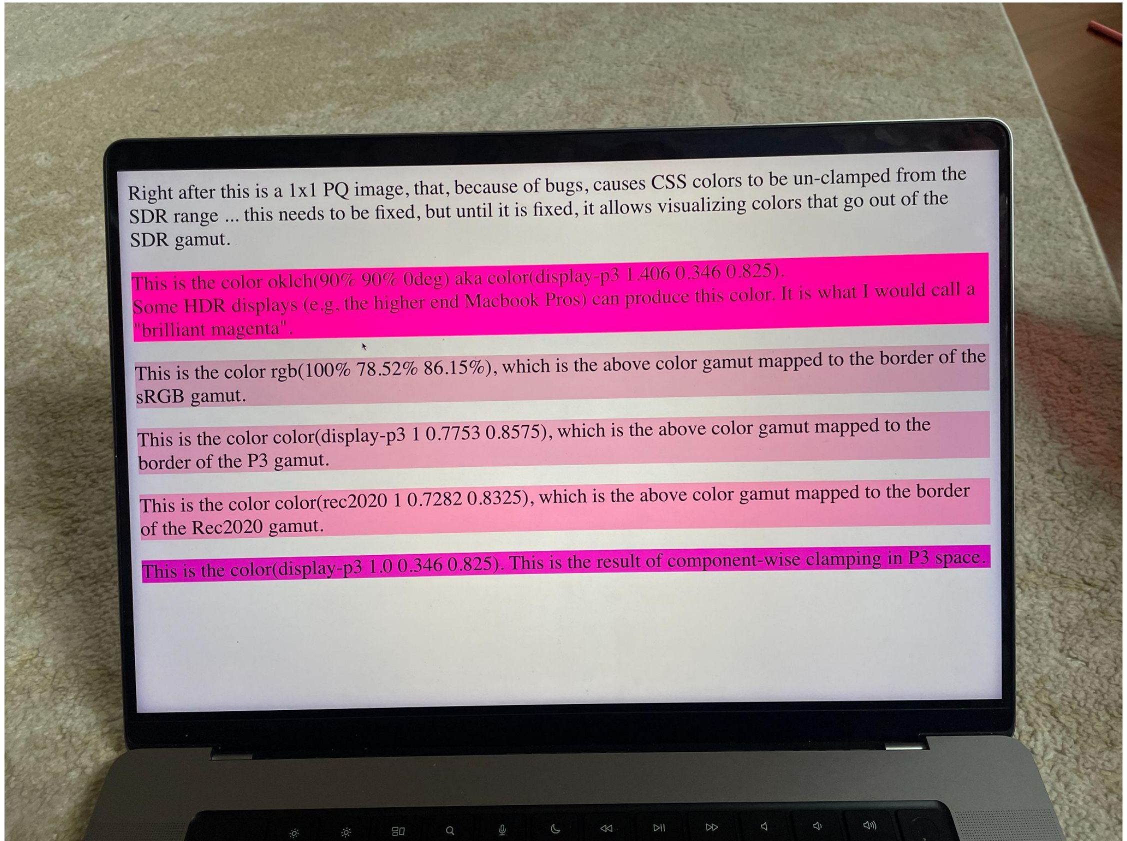
This is the color rgb(100% 78.52% 86.15%), which is the above color gamut mapped to the border of the sRGB gamut.

This is the color color(display-p3 1 0.7753 0.8575), which is the above color gamut mapped to the border of the P3 gamut.

This is the color color(rec2020 1 0.7282 0.8325), which is the above color gamut mapped to the border of the Rec2020 gamut.

This is the color(display-p3 1.0 0.346 0.825). This is the result of component-wise clamping in P3 space.

What should `oklch(90% 90% 0deg)` be?



Right after this is a 1x1 PQ image, that, because of bugs, causes CSS colors to be un-clamped from the SDR range ... this needs to be fixed, but until it is fixed, it allows visualizing colors that go out of the SDR gamut.

This is the color oklch(90% 90% 0deg) aka color(display-p3 1.406 0.346 0.825).
Some HDR displays (e.g, the higher end Macbook Pros) can produce this color. It is what I would call a "brilliant magenta".

This is the color rgb(100% 78.52% 86.15%), which is the above color gamut mapped to the border of the sRGB gamut.

This is the color color(display-p3 1 0.7753 0.8575), which is the above color gamut mapped to the border of the P3 gamut.

This is the color color(rec2020 1 0.7282 0.8325), which is the above color gamut mapped to the border of the Rec2020 gamut.

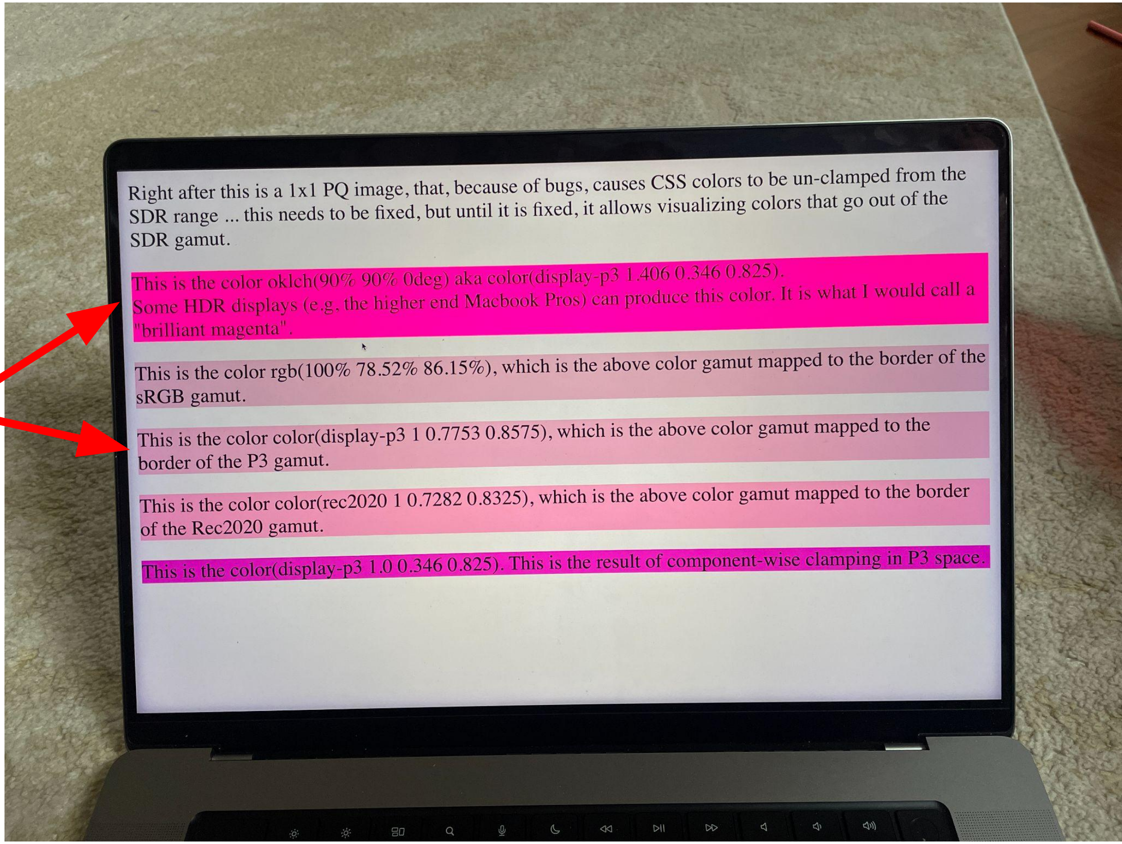
This is the color(display-p3 1.0 0.346 0.825). This is the result of component-wise clamping in P3 space.

What should `oklch(90% 90% 0deg)` be?

An author who specifies `oklch(90% 90% 0deg)`…

will see this today

but actually specified this, and one day will see that



Right after this is a 1x1 PQ image, that, because of bugs, causes CSS colors to be un-clamped from the SDR range ... this needs to be fixed, but until it is fixed, it allows visualizing colors that go out of the SDR gamut.

This is the color oklch(90% 90% 0deg) aka color(display-p3 1.406 0.346 0.825). Some HDR displays (e.g, the higher end Macbook Pros) can produce this color. It is what I would call a "brilliant magenta".

This is the color rgb(100% 78.52% 86.15%), which is the above color gamut mapped to the border of the sRGB gamut.

This is the color color(display-p3 1 0.7753 0.8575), which is the above color gamut mapped to the border of the P3 gamut.

This is the color color(rec2020 1 0.7282 0.8325), which is the above color gamut mapped to the border of the Rec2020 gamut.

This is the color(display-p3 1.0 0.346 0.825). This is the result of component-wise clamping in P3 space.

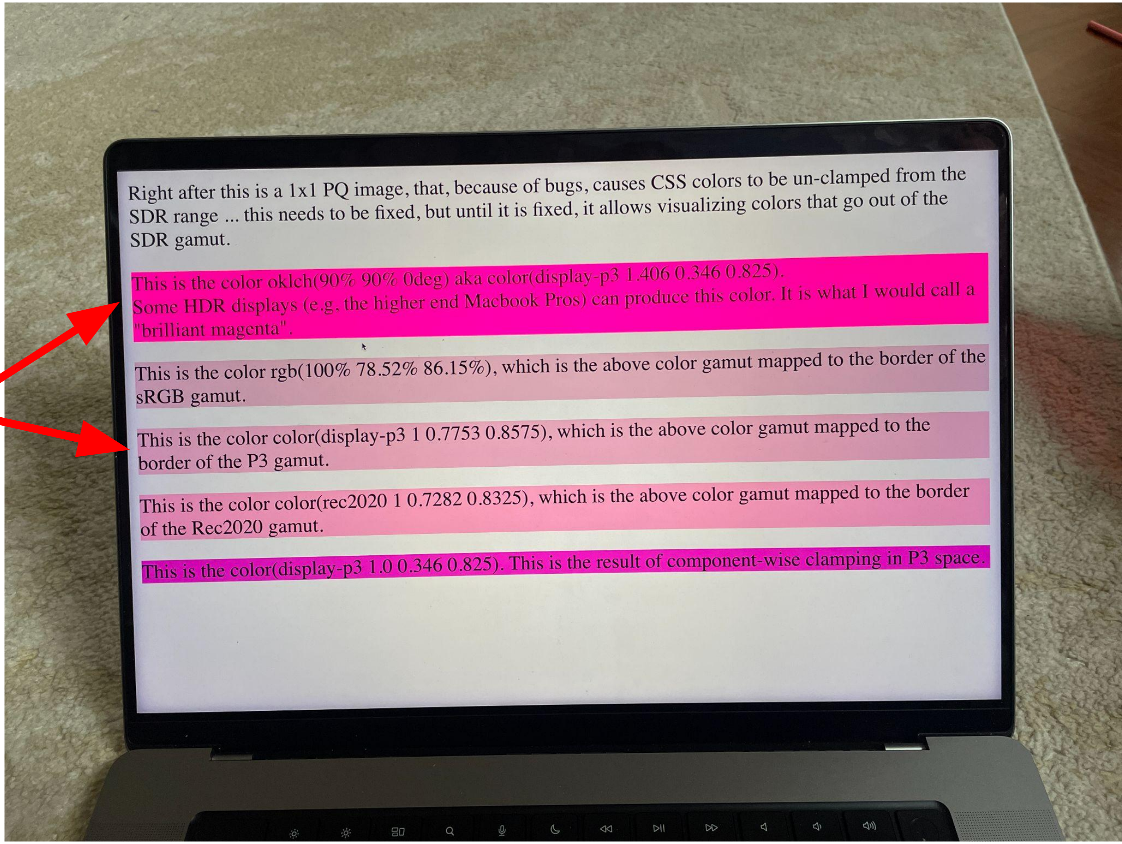
What should `oklch(90% 90% 0deg)` be?

An author who specifies `oklch(90% 90% 0deg)…`

will see this today

but actually specified this, and one day will see that

this will be a problem for us in the future



Right after this is a 1x1 PQ image, that, because of bugs, causes CSS colors to be un-clamped from the SDR range ... this needs to be fixed, but until it is fixed, it allows visualizing colors that go out of the SDR gamut.

This is the color oklch(90% 90% 0deg) aka color(display-p3 1.406 0.346 0.825).
Some HDR displays (e.g. the higher end Macbook Pros) can produce this color. It is what I would call a "brilliant magenta".

This is the color rgb(100% 78.52% 86.15%), which is the above color gamut mapped to the border of the sRGB gamut.

This is the color color(display-p3 1 0.7753 0.8575), which is the above color gamut mapped to the border of the P3 gamut.

This is the color color(rec2020 1 0.7282 0.8325), which is the above color gamut mapped to the border of the Rec2020 gamut.

This is the color(display-p3 1.0 0.346 0.825). This is the result of component-wise clamping in P3 space.

CSS should not dictate gamut mapping

CSS should not dictate gamut mapping

We’ve established we should never have to gamut map >Rec2020 to <sRGB

There are lots of tools that already do this (display profiles, MDCV metadata)

Just use those!

Proposed resolutions

Proposed resolutions

Add non-normative text to advise authors to not specify colors that:

- do not physically exist
- their display cannot produce (weaker)

Proposed resolutions

Add non-normative text to advise authors to not specify colors that:

- do not physically exist
- their display cannot produce (weaker)

Remove gamut mapping from its current place in the spec

Proposed resolutions

Add non-normative text to advise authors to not specify colors that:

- do not physically exist
- their display cannot produce (weaker)

Remove gamut mapping from its current place in the spec

Change spec definitions of colorspace to:

- bake gamut mapping to Rec2020 into `oklab` and `oklch`
- consider baking gamut mapping into `lab` and `lch`
- consider enforcing [0,1] parameter range for RGB spaces