New Things We Love And More Things We Want In CSS

Florida DrupalCamp 2023

Saturday, February 18th, 2023

Aubrey Sambor

Senior Front-End Developer

Adam Varn

Front-End Developer

Who We Are

Aubrey Sambor

Senior Front-End Developer @ Lullabot

- Over 20 years experience writing HTML and CSS
- Worked in Drupal since 2009
- Loves nerding out on CSS and a11y
- Lives in western Massachusetts
- First time FLDC attendee!

Mastodon - labyrinth.social/@starshaped
Twitter - twitter.com/starshaped

Adam Varn

Front-End Developer @ Lullabot

- Worked in Drupal since 2009
- Hot Sauce Design & Development from 2010-2021
- Love working on accessibility and front-end design challenges
- Live in Tampa
- Co-organizer of FLDC
- Attended every FLDC but two(!)

Mastodon - drupal.community/@hotsauce
Twitter - twitter.com/hotsaucedesign

New(ish) CSS Awesomeness

```
@container
:not()
:has()*
:is()
:where()
line-clamp
@layer
CSS Logical Properties
```

Now supported in all major browsers!
(IE 11 doesn't count)

*Behind a flag in Firefox for now

More Things We Want!

```
text-wrap: balance
@container style()
color-contrast()
CSS Subgrid
Better comments!
```

CSS Logical Properties

Or: Directions? We don’t need no stinkin’ directions!

CSS Logical Properties

"CSS Logical Properties and Values is a module of CSS introducing logical properties and values that provide the ability to control layout through logical, rather than physical, direction and dimension mappings."

MDN Article

- Uses `block` (vertical) and `inline` (horizontal) as terminology
- Applies to sizing, margins, padding, floats*, positioning & borders
- Fully adopted in 2017 by all browsers, useful for `direction` and `writing-mode`

*only supported by FF for now

CSS Logical Properties

Positioning

`top` ➡️ `inset-block-start`

`bottom` ➡️ `inset-block-end`

`left` ➡️ `inset-inline-start`

`right` ➡️ `inset-inline-end`

Width & Height

`width` ➡️ `inline-size`

`height` ➡️ `block-size`

`max-width` ➡️ `max-inline-size`

`min-height` ➡️ `min-block-size`

CSS Logical Properties

Margins

`margin-top` ➡️ `margin-block-start`

`margin-bottom` ➡️

`margin-block-end`

`margin-left` ➡️

`margin-inline-start`

`margin-right` ➡️

`margin-inline-end`

Padding

`padding-top` ➡️ `padding-block-start`

`padding-bottom` ➡️ `padding-block-end`

`padding-left` ➡️

`padding-inline-start`

`padding-right` ➡️ `padding-inline-end`

CSS Logical Properties

Why do CSS Logical Properties matter?

- Languages other than English
- Agnostic terminology
- Works the exact same way as non-logical properties
- Future you will thank you!

Codepen Demo

@container queries

Or: Change a component without using a media query

@container queries

"[@container] provides a way to isolate parts of a page and declare to the browser these parts are independent from the rest of the page in terms of styles and layout."

MDN Article

- Defined with `container-type` and `container-name` properties.
- `container-type`:

  `size`: based on inline (width) and block (height) dimensions of container.

  `inline-size`: based only on the inline size (width).

  `normal`: not used for size queries but needed for reference (default).

- `container-name`

  - Used to define the name of the container for use later on.

@container queries

```
.layout-container {
      container-type: inline-size;
      container-name: desktop-container;
}

.call-to-action {
  width: 33.333%;
}

@container desktop-container (min-width: 768px) {
  .call-to-action {
    width: 50%;
  }
}
```

Codepen Demo

:is()

Or: This :is() the best selector ever!

:is()

*"The **:is()** CSS pseudo-class function takes a selector list as its argument, and selects any element that can be selected by one of the selectors in that list"*

MDN Article

- Useful for simplifying CSS selectors to a single line
- Also takes on specificity of its most specific selector
- Accepts a forgiving selector list
  - `:is(.cats, :kittens)` will select `.cats`, though `:kittens` is invalid
- Doesn't select pseudo-elements, so `:is(::before, ::after)` won't work

:is()

```css
/* Long, unwieldy list of selectors. Bad! */
main h1,
main h2,
main h3,
main h4,
main h5 {
  color: rebeccapurple;
}

/* The same code written in just one line. Good! */
main :is(h1, h2, h3, h4, h5) {
  color: rebeccapurple;
}
```

Codepen demo

:where()

Or: Sneaking in with no specificity

:where()

*"The **:where()** CSS pseudo-class function takes a selector list as its argument, and selects any element that can be selected by one of the selectors in that list."*

MDN Article

- Just like `:is()`, but `:where()` always has zero specificity
- But what does this mean, exactly?

:where()

```
:is(header.testing-is, section.testing-is, footer.testing-is) a {
  color: hotpink;
}

:where(header.testing-where, section.testing-where, footer.testing-where) a {
  color: rebeccapurple;
}

footer a {
  color: yellowgreen;
}
```

Codepen demo

:not()

Or: Not just another boring selector

:not()

*"The **:not()** CSS pseudo-class represents elements that do not match a list of selectors. Since it prevents specific items from being selected, it is known as the negation pseudo-class."*

MDN Article

- Has been supported since IE9!
- Selector argument list not supported by IE
  - `:not(.cats, .kittens)` needs to be written as

    `:not(.cats):not(.kittens)` in IE
- Does not accept a forgiving selector list like `:is()` and `:where()`
  - `:not(.cats, :kittens)` will *not* be valid.

:not()

```css
/* Instead of writing this ... */
li {
  margin-bottom: 1.5em;
}

li:last-child {
  margin-bottom: 0;
}

/* You can just write this! */
li:not(:last-child) {
  margin-bottom: 1.5em;
}
```

Codepen demo

:has()

Or: A pseu-per pseudo selector

:has()

“[:has()] represents an element if any of the relative selectors that are passed as an argument match at least one element when anchored against this element.”

MDN Article

- Allows the ability to select an element’s parent (finally!)
- `div:has(img)` will select all `div`s that have an image
- Takes on the specificity of its most specific selector
    - `div:has(#img__id)` has higher specificity than `div:has(img)`

:has()

```
.grid__item {
  border: solid 5px #ccc;
  border-radius: 12px;
  padding: 16px;
}

.grid__item:has(img) {
  border-color: hotpink;
  padding: 0;
  overflow: hidden;
}
```

Codepen demo

@layer

Or: If you are cold, put on a @layer

@layer

*“The **@layer** CSS at-rule is used to declare a cascade layer and can also be used to define the order of precedence in case of multiple cascade layers.”*

MDN Article

- `@layer` allows you to set styling no matter where it is in the cascade
- Works either with set layer names (i.e. `@layer toolbar`) or unnamed
- Layers can be imported within each other or as separate files
- Supported in all browsers! 🎉

@layer

```
/* Layers can be anonymous with no name */
@layer {
  * {
    font-family: arial, sans-serif;
  }
}

/* Or you can call them with a name too. The order declaration
  of the layers determines the order in which they render and
  their precedence. Below, the stuff in "secondary-layout" gets
  precedence because it's last, even though the "main-content"
  styling is set below it. Flip the name order to see the effect!
*/

@layer main-content, secondary-content;

@layer secondary-content {
  p {
    color: rebeccapurple;
  }
}
@layer main-content {
  p {
    color: red;
  }
}
```

Codepen Demo

line-clamp

Or: -webkit-line-clamp to all the OGs

line-clamp

*"The **-webkit-line-clamp** CSS property allows limiting of the contents of a block to the specified number of lines. It only works in combination with the display property set to -webkit-box or -webkit-inline-box and the -webkit-box-orient property set to vertical."*

MDN Article

- Shortens text to a specified line limit, with ellipsis at the end
- `line-clamp` also works and is meant to replace `-webkit-line-clamp`
- Text is still present but visually hidden - works in all browsers despite prefix!

line-clamp

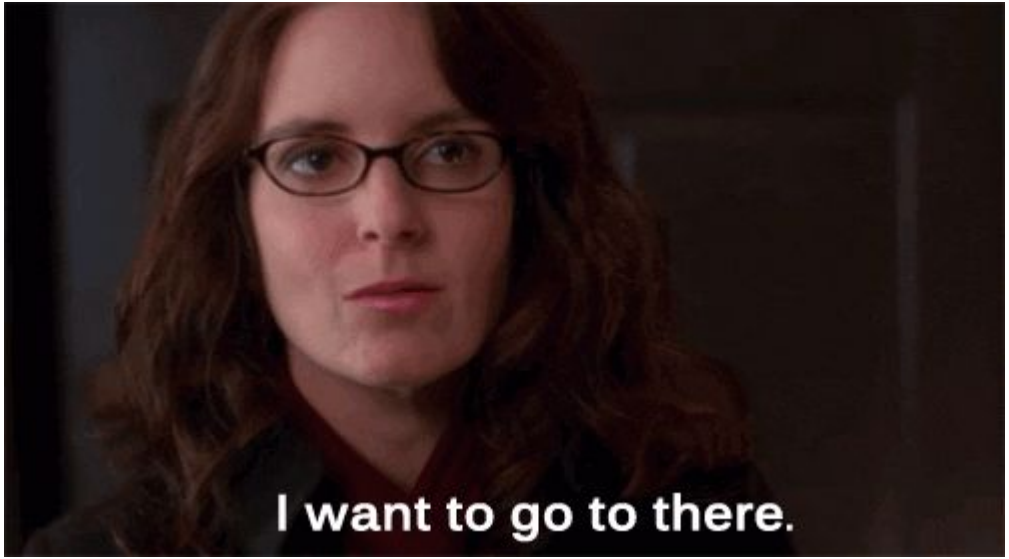
```
.my-text-block p:nth-of-type(3) {
  /*  These four lines MUST be present  */
  display: -webkit-box;
  -webkit-box-orient: vertical;
  -webkit-line-clamp: 3;
  overflow: hidden;

  /* line-clamp also works
  line-clamp: 4; */

}
```

Codepen Demo

The Things We Still Want



text-wrap: balance

“Same as wrap for inline boxes. For block containers that establish an inline formatting context, line breaks are chosen to balance the remaining (empty) space in each line box, if better balance than wrap is possible.”

W3.org Spec

- Would make each line of text (approximately) the same length
- Would work with fluid columns (i.e. grid and responsive design) when lengths are not known
- Still in spec mode, no ETA on adoption but is in “Intent To Prototype” for Chrome

text-wrap: balance

```
Checkout this really long line of text, boy does it
go on for a while huh? I wonder when it will end? Who
knows, but it sure is long!
```

would become this:

```
Checkout this really long line of text, boy
does it go on for a while huh? I wonder when
it will end? Who knows, but it sure is long!
```

Subgrid

“Feature of the CSS Grid Layout Module Level 2 that allows a grid-item with its own grid to align in one or both dimensions with its parent grid.”

caniuse.com

- Supported in Firefox and Safari only, still in development in Chromium browsers
- To use, add `grid-template-columns/rows: subgrid` on a grid item
    - The grid item also needs to be set to `display: grid`!

Subgrid

```css
.grid {
  /* Set grid on main container */
  display: grid;
}


.grid__item {
  /* Set subgrid on item within the grid! */
  grid-template-rows: subgrid;

  /* Don't forget to set the item display to grid ... */
  display: grid;
}
```

Codepen demo

color-contrast()

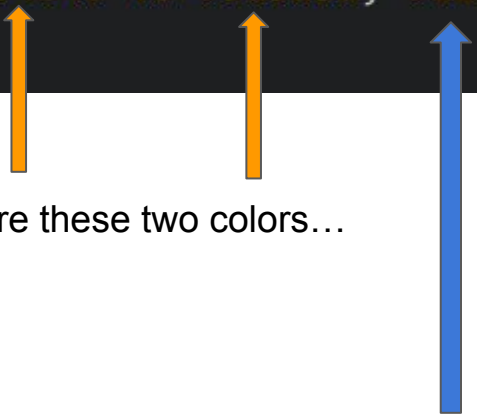
“The contrast-color() functional notation identifies a sufficiently contrasting color against a specified background or foreground color without requiring manual computation.”

W3.org Draft Spec

- Compares two set colors vs. a defined background color
- Would save a lot of time addressing a11y issues, useful with CSS variables
- Still a W3C CSSWG draft, no browsers support it yet 😕

color-contrast()

```
.header-color {
  color-contrast: (#404040 vs #5a5a5a, #111);
}
```



Would compare these two colors…

…against this color, and pick the one with the higher contrast level automatically to set a `color` or `background-color` property, for example.

// comments like this

Seriously, why are we stuck with `/* */` as our only option?

More CSS goodness

- CSS margin-trim
  https://developer.mozilla.org/en-US/docs/Web/CSS/margin-trim
- New viewport units (svh, dvh, lvh)
  https://www.matuzo.at/blog/2022/100daysof-day38/
- CSS style queries
  https://ishadeed.com/article/css-container-style-queries/
- All the CSS wishlists!
  https://chriscoyier.net/2023/02/14/2023-css-wishlists/

Tada!