CSS: It was twenty years ago today — an interview with Håkon Wium Lie

Twenty years ago today, Opera's CTO Håkon Wium Lie published Cascading HTML style sheets – a proposal. If Paul McCartney were a web developer, and writing 'Sergeant Pepper's Lonely Hearts Club Band' today, he would almost certainly write:

It was twenty years ago today
That Håkon wrote a doc to say
That if the Web's gonna last a while
Then we need a way to define style.
So may I introduce to you
a way to add visual treats:
It's Sergeant Håkon's Cascading Style Sheets!

However, when we went round to Paul's house to ask him to sing this for us, he declined and set his guard dogs, `FontTag` and `Bgcolor`, on us. So, instead, to mark this occasion, Bruce sat down with Håkon to ask a few questions about the past, present and future of CSS.

Opera's CTO Håkon Wium Lie

CSS was conceived twenty years ago. Are you happy about how your baby, child and teenager has turned out?

Yes, I'm very happy with CSS. CSS is a cornerstone web specification and web pages are more beautiful as a result of CSS being there. The fact that HTML is still alive and well is also a testimonial to the success of CSS. As with all youngsters, however, there is room for improvement.

You've been quoted as saying you proposed CSS "to save HTML". Please explain.

HTML would have been very different if CSS had not appeared. Authors who came to the web from a desktop publishing background were baffled by the lack of `<color>` and `<font>` tags. Including myself. In my first real publication on the web, I resorted to making images with text in it — you can see it in this publication from 1993. If this development had continued, the web would have become a giant fax machine where pictures of text would be passed along. This would have been terrible for blind users and search engines alike. CSS was proposed to prevent this development by giving authors a way to express their designs without adding new HTML tags.

In the first proposal, there was a percentage of influence specifier, e.g. `h1.font.size = 24pt 100%`, described as follows:

The percentage at the end of the line indicates what degree of influence that is requested (here 100%). If this is the initial style sheet (i.e. the one under user control), this request can be fulfilled, i.e. all headline elements will be rendered using Helvetica. If the statement comes in a later style sheet, any unclaimed influence is granted.

Why was this dropped?

The mechanism you refer to was there to try combine the needs and preferences of both authors and readers. It was inspired by an idea from the MIT Media Lab: TVs in the future would not have controls for brightness and color, but for sex and violence, or left-wing and right-wing, perhaps. The CSS proposal was to have sliding scale where the author was fully in charge in one end of the scale, and the user was fully in charge in the other end of the scale. In between, the browser would try to mix requests to make everyone happy. This works well for certain properties (like `font-size`) but is hard to do for others (like `font-family`). In his first response to the CSS proposal, Bert criticized this idea:

The idea that two designs can be averaged to come up with an intermediate style seems utterly wrong to me. What happens when my blue-on-yellow style is combined with somebody else's yellow-on-blue? Do I get green-on-green? Or who wants to look at a page with Avant-garde titles over Helvetica paragraphs?

I responded:

Some attributes mix better than others. A typical use of "weighted average" is to soften the author's attempt to be distinct, e.g. the suggested font sizes – while the user still gets the message. One doesn't have to use this feature, but while "100%" is equal to a binary "1" there is no going back from a binary syntax. In general, I think computer interfaces are much too binary.

Of course, Bert was right (he is, almost always) that the proposal created more problems than it solved, and the mixing was dropped.

Ditto the JS-style dot syntax. Why did it change?

It's interesting you call it JS-style syntax. When I first proposed CSS, JavaScript was not around so I couldn't borrow it from there. Rather, the syntax was inspired by X resources from the X11 Window System, another inspirational project coming out of MIT.

The reason for changing the CSS syntax from `font.size` to `font-size` was twofold. First, the hyphen makes it look more like written English, which improves human readability. Second, DSSSL and DSSSL-Lite used hyphenated property names. James Clark, who wrote the first draft of DSSSL-Lite, participated in the first W3C workshop on style sheets, and Bert and I borrowed the hyphen from DSSSL. On the downside, the hyphen represents minus in math, which sometimes confuses parsers.

How did Bert Bos get involved, and how did you work together?

Bert Bos reviewed my initial proposal. His background and focus was a bit different from mine, but when he wrote up his own proposal we quickly realized that two proposals could be combined into one. At that point, the web project was being kicked out of CERN and W3C was formed. I was starting up the European branch of W3C at INRIA and Bert was hired immediately. Most of CSS1 was hammered out on a whiteboard in Sophia-Antipolis in July 1995. Bert is still working for W3C in Sophia-Antipolis. Whenever I'm struggling with a difficult technical problem, I wish Bert and the whiteboard were there.

When listing people who made CSS possible, I must also mention Thomas Reardon and Chris Wilson of Microsoft. Thomas was the program manager for Internet Explorer (IE) who early caught onto the idea of adding style sheets to the web. Chris Wilson was the programmer who added CSS to IE3. IE3's implementation was far from the standard, but one must remember that it was released before CSS1 was finished. Simon Daniels (also of Microsoft) wrote some impressive demos in IE3, and together they committed a major software company to support an emerging standard.

Were there any competing proposals? Why was yours better?

There were a dozen or so proposals for style sheets languages to be used. However, not all of them were suitable due to the characteristics of the web. For example, browsers use progressive rendering to display documents, and they must handle situations where style sheets are not accessible. There's a wide range of web devices out there and you can't write a style sheet for each one. So, style sheet languages must express designs that are scalable and responsive. It's a fascinating area of study and I've written a PhD Thesis, which compares the approach taken by the various style sheet proposals.

Naturally, I'm biased in the second part of your question. But I believe CSS has several notable features which makes it especially suitable for web use: cascading, pseudo-classes and pseudo-elements, forward-compatible parsing rules, support for different media types, strong emphasis on selectors, and the amazing `em` unit.

What's the biggest mistake you (and Bert) made?

I devote a chapter in my PhD thesis to problems in CSS. There are some, even self-inflicted ones. But the biggest problem CSS1 experienced was not in its design (which is pretty good, if you ask me), but rather in the initial implementations. Jeffrey Zeldman described the situation:

If Netscape 3 ignored CSS rules applied to the `<body>` element and added random amounts of whitespace to every structural element on your page, and if IE4 got `<body>` right but bungled padding, what kind of CSS was safe to write? Some developers chose not to write CSS at all. Others wrote one style sheet to compensate for IE4's flaws and a different style sheet to compensate for the blunders of Netscape 4.

Microsoft and Netscape both deserve some blame, but we — me, Bert, W3C — could have avoided many problems by producing a test suite along with the CSS1 specification. The first real CSS test appeared in October 1998 when Todd Fahrner published his Acid test. The test was creative, visual, and developers could immediately see whether they passed the test or not. In the beginning, no browsers passed. But then, with a strong push from the CSS community, things started to improve. The WaSP played a key role in assuring support for standards. The Opera browser also played a role by showing it was actually possible to implement CSS correctly. When I realized how solid Opera's CSS implementation was — much better than Microsoft's or Netscape's — I joined Opera.

One of Opera's claims to fame before I joined was that the browser would fit on a floppy disk (which is 1.44 MB). "Fits on a floppy" was a great slogan in those days. When CSS was added, a few more bytes were needed and Opera would no longer fit. "Almost fits on a floppy" doesn't quite have the same ring to it.

The original Acid test inspired Acid2 and Acid3, which use the same formula: a visual and demanding web page which tests a wide range of features. The development of Acid2 was triggered by an open letter from Bill Gates, where he praised interoperability. Making IE7 follow the CSS standards seemed like a natural follow-up item, and Microsoft was challenged by Acid2. In fairness, other browsers also had issues, and Acid2 exposed bugs in all of them. Microsoft ignored

Acid2 in IE7, but — magic sometimes happens — IE8 supported it perfectly. As do all current browsers.

Why did you decide on the box model whereby margin, padding, border are added to declared width rather than the IE5 `box-sizing: border-box` model?

There are good use cases for both models, I believe. If you want an image to stretch out to fill the whole content box, the original CSS box model is the one to use. However, if it's important that padding and borders do not extend outside a certain area, the IE5 model is better. Personally I think there are more use cases for the CSS box model, but some people that I respect highly think otherwise. The conflict has been gracefully resolved by the addition of the `box-sizing` property which all browsers now support.

I've always disliked absolute positioning. Am I wrong? How did it come about in the spec?

Your question takes me back to some heated debates in 1996. The short story is that Microsoft proposed absolute positioning in a draft called CSS Regions: Absolute Positioning and Z-Ordering (discussions took place on W3C member-only lists, I'm afraid; the closest public document is WD-positioning). Several members of the newly formed CSS Working group had reservations, and Bert and I wrote up a simplified counter-proposal. Our proposal got rid of the `position` property (`display` was used instead) and only described relative positioning (which would give us time to think through absolute positioning). Microsoft, however, had already implemented their proposal and were reluctant to remove functionality. In the end, the only material changes were to add the `right` and `bottom` properties (to balance `left` and `top`), and to add `position: fixed`. This became part of CSS2.

Like you, I have never become comfortable with absolute positioning. That being said, absolute positioning has found its place on the web and I use it from time to time to achieve things that would otherwise be hard, if not impossible, to encode.

I've heard people say that you shouldn't use floats for layout, as they "weren't intended" for that — they were just to wrap text around images. Does the intention matter, if they work?

Having text wrap around images is one of the basic layout techniques. For sure, floats should be used for layout. One area I'd like for CSS to progress into is paginated onscreen presentations. When you paginate content, floats become even more useful because you can float elements to the top and bottom of the screen.

If you could wave a magic wand, which bit of current CSS would you banish from the world, and what would you magically add and implement everywhere?

I'd banish browser-version-specific code like: `<!--[if lt IE 7 ]>`. While technically not expressed in CSS, these kinds of "comments" should not be necessary and they lower the standards of the web.

The second part of your question is even more interesting: what parts of CSS should be magically implemented everywhere? In 2006, I would have said web fonts. In 2007, I would have said the `<video>` element (straying into HTML-land for a while). Both of these are now implemented in all browsers.

In 2011, seeing that many apps used pages (and not scrollbars) to create compelling presentations, I started advocating for web pages to become real pages. The idea is that the style sheet would trigger paged mode so that content is split into pages. Users would navigate from one page to the next with gestures, or perhaps with `PageUp` and `PageDown`. I want it to be possible, even easy, to create ebook-readers in browsers. For this to happen, a bit of magic would be helpful. Could you sprinkle some fairy dust on the pillows of all browser vendors, please?

But CSS is not just for browsers. Bert and I wrote a book on CSS and in 2005, for the third edition, we wanted to use CSS itself for formatting. The browsers were not up to the job, but then Prince came along. Michael Day and his colleagues in Melbourne made a terrific product which was able to create a beautiful PDF document from HTML and CSS which we could send to the printer. In order to them to fix my favorite bugs, I joined the board of directors. Prince has since been used to format hundreds of books. We will still have some paper books in the future, I believe. And they will be made from HTML and CSS.

Why do we still not have a method of laying out pages in CSS that doesn't require a Doctorate in Rocket Surgery? (I'm looking at you, Flexbox…)

Layout is complex, and layout on the web is even more complex due to pages being displayed on so many different devices. CSS has several mechanisms for laying out content, including absolutely positioned elements, floats, multi-column layout, and CSS tables. The interaction between them can be complex, but I don't think any doctorate is needed. I must admit to not having used Flexbox much, though.

What's your opinion on pre-processors like SASS and LESS? Does CSS have anything to learn from them?

Yes, pre-processors make a lot of sense. We should probably take the five or so most popular features from pre-processors and add them to CSS itself. My own favorites would be nested selectors, and single-line comments (starting with `//`). When CSS turns 50 I'll tell you why they were not part of CSS from the beginning.

You're one of the members of the WHATWG. How did that come about?

WHATWG was formed when it became clear that W3C would abandon work on HTML and instead focus efforts on compound documents based on XHTML, XForms, SMIL and SVG. For browser makers, HTML was much too important to abandon. Therefore, Ian Hickson, who, at the time, worked in my group at Opera, set up WHATWG to continue working on the web as we know it. Also, we were concerned about Microsoft's XAML, which had a thin layer of XML on top of a proprietary application language. The focus of WHATWG is therefore on applications, rather than documents. Ian continues to do an amazing job as the editor of the HTML standard.

You're the father of CSS, yet you've recently published some specs under the WHATWG domain rather than the CSS WG in the W3C. Why?

Indeed, CSS Figures and CSS Books are work items of WHATWG. There are some important benefits to publishing the specification in WHATWG. The "living standard" model allows quick updates with little overhead; in the past it was difficult to publish W3C Working Drafts in this area. WHATWG's goal of keeping the specifications a little ahead of the implementations but not so far ahead that the implementations give up is one that I strongly believe in.

Finally, is CSS still up to the job? Or should we try to move to another model — for example, Grid Style Sheets?

Ethan Munson and Philip M Marden wrote in 1999 that "style sheet languages are terribly under-researched". This remains true today, and efforts to research and improve style sheet languages must be encouraged.

GSS is an interesting example which adds the "Cassowary" constraint solver to the style sheet mechanism. In my initial CSS implementation in 1995 I used the "SkyBlue" constraint solver to resolve conflicting style sheet statements. (By the way, "SkyBlue" and "Cassowary" were both developed at the University of Washington.) Having a constraint solver allows you to express relationships between arbitrary elements and have conflicts resolved automagically. However, things can get complex when elements disappear and new ones arrive, like they do through DOM operations. Circular dependencies must also be handled gracefully. Therefore, the idea of allowing CSS to express layout constraints between any elements were dropped at an early stage.

In the past, adding support for a new specification required you to convince all browser makers to devote precious developer time. This raised the bar, perhaps uncomfortably high. These days, it's possible to extend browsers by way of JavaScript libraries. This makes it easier to experiment and perform some of the research style sheets deserve.

Returning to your question: is CSS still up for the job? I think so. I don't see anything on the horizon that makes me think otherwise. New ideas will come along, but they will extend CSS rather than replace it. I believe that the CSS code we write today will be readable by computers 500 years from now.

Thank you. Happy CSS Birthday!

Bruce Lawson

Published on 10 October 2014 in Articles. Edit this article on GitHub. Licensed under a Creative Commons Attribution 3.0 Unported license.

css