Design
Arts
Médias

Designing with Abstractions: CSS and the Case of Masonry Layouts

Julie Blanc

2025-06-30

Julie Blanc est titulaire d'un doctorat en ergonomie (psychologie) et design (Université Paris 8 / Artec / EnsadLab, juin 2023). Sa thèse porte sur l'utilisation des technologies web pour la publication imprimée et le développement des communautés de pratiques qui y sont associées. Elle a fait partie de l'équipe qui développe paged.js et est spécialiste du CSS print (web-to-print). Co-fondatrice de Studio Variable, elle travaille majoritairement des projets de publications mélant code et design graphique. Elle est actuellement collaboratrice scientifique à la HEAD Genève pour le projet de recherche « WYSIWYG, An Investigation in the updake of graphic design software in Switzerland and France, 1980 – today ».

Abstract

This article analyzes the debates around integrating the Masonry layout into CSS, focusing on the process of abstraction and standardization. It examines CSS as both a design object and a formal system, shaped through conceptual debates, technical implementation concerns, and interface-centered considerations. The article concludes by highlighting how abstraction is collectively negotiated to form a common foundation for the practice.

Keywords

CSS, Masonry layout, abstraction, standardization, design

Résumé

Cet article analyse les débats entourant l'intégration de *Masonry layout* dans CSS, en se concentrant sur le processus d'abstraction et de standardisation. Il examine CSS à la fois comme un objet de design et comme un système formel, façonné par des débats conceptuels, des préoccupations liées à la mise en œuvre technique et des considérations centrées sur l'interface de code pour les designers et développeurs. L'article conclut en soulignant comment l'abstraction est négociée collectivement afin de constituer une base commune pour la pratique.

Mots-clés

CSS, Masonry layout, abstraction, standardisation, design

Introduction

For some years now, graphic design has had a lot to do with code. Since the appearance of the web, Cascading Style Sheets (CSS) have become central to shaping visual experiences online. Yet, unlike traditional desktop publishing tools often based on direct manipulation metaphors and WYSIWYG (What You See Is What You Get) interfaces, CSS is fundamentally a *language*, i.e., a formal system requiring designers to engage with abstraction and logical rules. As American UX-designer A. J. Kandy writes (with great clarity):

> *Still, today, the only way to really design* for *the web,* on *the web, with precise control, is via markup languages and programming code. Thinking like the machine, to get the machine to do what you want*[1].

The layout system in CSS is one of the most challenging and emblematic parts of this logic, demanding a translation of visual intent into declarative rules. This article explores these dynamics through a specific case study: the recent and intensive debates surrounding the integration of Masonry layout — a type of grid layout — into the CSS standard. This specific standardization effort provides a rich case where the challenges of abstraction and the socio-technical processes

of standardization converge.

Analyzing this case serves a double purpose. First, it allows us to understand CSS itself *as a design object*, shaped through historical, technical, and practical choices. Second, it brings into focus the fundamental process of abstracting visual layout into a formal language. We hope to reveal how such abstractions are negotiated and solidified at a collective level, shaped within and for the field of graphic web design.

This approach echoes the idea that coding is not merely a medium *for design*, but a space *of design* itself. As Katherine N. Hayles and later Adrian Mackenzie — drawing on Judith Butler's theory of performativity — suggest, code is performative in a strong sense: it brings into being what it describes. The abstraction here is not an intermediary tool but the very substance of design. The code *is* the designed object, and the act of coding is an act of shaping material.

To develop this argument, the article proceeds as follows. We first delve into the concept of abstraction, in computer science and graphic design. Following this conceptual part, we provide a brief history of CSS layout to contextualize the specific challenges addressed by modern layout methods. We then outline the socio-technical processes of standardization within the World Wide Web Consortium (W3C), detailing the methodology used to analyze the Masonry layout debates. The core of the paper presents an in-depth analysis of the Masonry layout case study, tracing its origins and dissecting the differents arguments deploy by differents actors (conceptual debates, technical implementation concerns, and interface-centered considerations). Finally, we conclude by synthesizing these findings to underscore how CSS functions as a designed object itself and how abstraction is collectively negotiated to form a common foundation for the practice.

Abstraction in (Graphic) Design and Computation

Since our aim is to explore abstraction within the framework of CSS, it is relevant to define this concept in two fields closely connected to it: computer science and graphic design.

In computer science, abstraction is a central and fundamental concept, often considered to be the most important mental tool for computer scientists[2]. It refers to the ability to conceptualize systems by omitting non-essential details and focusing on relevant structural features. This allows the designer or programmer[3] to work across levels of detail —from general behavior to implementation specifics – and to construct reusable patterns that encapsulate common solutions.

> *Fundamentally, abstraction is the process of focusing on general concepts or the "big picture"—seeing the forest, not just the individual trees. (…) For example, simplifying a problem by overlooking non-crucial details in its description helps focus on the computational essence of the problem. Ignoring details can also be expressed as generalization or as distinguishing between "what" and "how". Generalization involves extracting common characteristics and* essence *from multiple instances while setting aside their distinguishing details*[4].

Concretely, it allows developers to manage complexity by working with simplified models, such as data types, functions, or components, without needing to understand their internal implementation.

In design, abstraction is more ambiguous. We focus on graphic design for this paper. Abstraction has traditionally operated through formal reduction — simplifying shapes and typographic forms, generalizing color palettes or grid layouts.

This understanding of graphic design as visual abstraction emerged with the introduction of phototypesetting into publishing processes. The graphic designer is the one who conceives the

layout of publications or the templates for visual communications. They conceived rule-based designs, later transmitted to others involved in the graphical chain: phototypesetting operators, “paste-up” people, and other graphic designers. In this sense, abstraction lies in the designer’s ability to anticipate how layouts will be implemented by others down the production line.

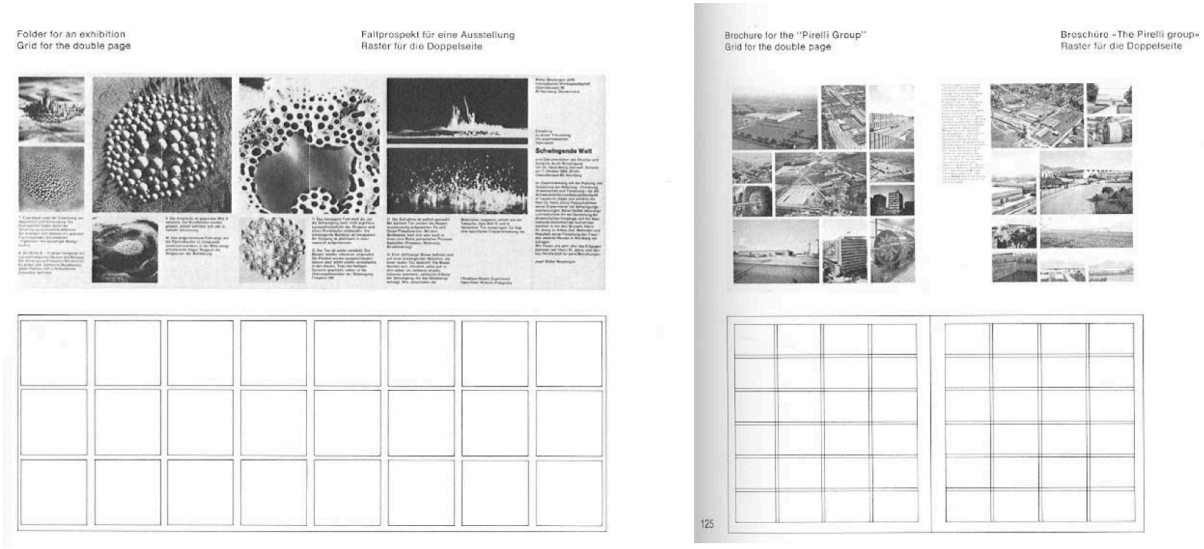

Figure. Example of grid by Josef Müller-Brockmann in his book Grid systems in graphic design (1981), pp. 124-125

However, the work of conceptualizing these layouts and templates was tactile and immediate, taking shape through direct manipulation — cutting, pasting, tracing, typesetting. Even the most conceptual gestures were ultimately rooted in the properties of physical tools and materials: grids were defined in millimeters, typography in points[5].

But as the tools of the graphic designer transitioned from analog to digital —from photocomposition to desktop publishing — the nature of abstraction transformed in kind. What was once done through immediate contact with matter now takes place through graphical interfaces that mediate the design process.

These interfaces, while often borrowing visual metaphors from analog tools— “paintboxes,” “toolboxes,” “cut” and “paste” — introduce a new distance between the designer and the form. Reflection on form is no longer shaped by the hand but by interaction with screens, menus, and layered compositions. The designer navigates a visual system where gestures are interpreted, translated, and constrained by the logic of the interface itself. However, the toolbox of Desktop Publishing softwares still represents an attempt at mapping designers’ traditions and existing mental models — particularly the model of the printed page[6].

The emergence of the web marked a pivotal inflection point. Where once the designer authored fixed compositions — intended for specific paper dimensions and a well-defined amount of content — they now deal with environments dependent on device, screen size, or user settings. The design of a website will not appear in the same way in these different contexts. To author the design, graphic designers may use CSS.

CSS, which stands for Cascading Style Sheets, is a domain-specific declarative language designed for styling HTML documents on the Web[7]. It describes how colors, fonts, and layouts are presented and allows web page presentation to be adapted to different devices, such as large and small screens[8]. CSS was designed to allow web content to be presented in different contexts, serving different user needs.

With CSS, the designer will have to describe generic rules for the behavior of elements, which will be instantiated in a specific way for a given display (screen size, type of machine, user settings). The designer, increasingly, writes instructions rather than directly shaping form. The rise of code

as a design tool introduces a new kind of abstraction — one that is not only rule-based but also procedural, generative, and variable. This shift expands the idea of “designing the object” to “designing the process by which the object emerges.” What was already true in some design practices[9] is here deeply entangled with the technology. Writing code becomes a way of modeling/shaping design, not just representing design.

This requires a new kind of thinking: not just about *how things look*, but about *how they behave*. Writing CSS means constructing a formal system of constraints and relationships between elements, allowing for variable instantiations rather than singular outcomes. That is, building a system that fluidly adapts across a continuous range of screen sizes — from narrow viewports to ultra-wide displays — rather than producing a single, fixed design for a predetermined format, like a print poster. Designers specify desired outcomes (e.g., “center this block”, “make all titles blue”) and the browser engine computes the final layout based on multiple contextual factors at render time.

> *Writing CSS is effectively setting up a system of constraints. You don’t tell the browser where to put every single element on the page; you tell it how much space to put between them and let it sort out where they belong. (…) There are too many variables to consider. The point of CSS is to make it so you don’t have to worry about them all. (…) This is the power of a declarative language [10].*

Unlike imperative programming languages[11] that articulate step-by-step procedures (JavaScript, C, or Python), CSS allows designers to define relationships, variables, and responsive patterns. With CSS, designers avoid micromanaging layout and instead focus on higher-level structural intent — specifying relationships like spacing, flow, or alignment while leaving the computation of layout to the browser.

In this way, CSS crystallizes a broader transformation in design practice: from the production of fixed surfaces to the scripting (“writing”) of dynamic spatial logic. To make it possible, CSS is a language intentionally designed to apply visual and behavioral rules to a set of contextual elements. This is where the logic of abstraction becomes most evident: CSS is built to generalize — to extract common structural principles from diverse cases while setting aside specific details. The definition of abstraction used in the field of computer science is applied here to a language used specifically for graphic design. In what follows, we will examine how this logic shapes the design of CSS itself, focusing on the domain of layout.

A Brief History of CSS Layout

Before moving forward, it’s worth stepping back to trace the development of the web and the CSS language.

The World Wide Web (W3 or the web) was designed as a device-independent platform. Tim Berners-Lee, his inventor, described it as a universal space, accessible regardless of hardware, software, language, or ability[12].

With this in mind, the web required a language that was simple, readable, and accessible on any platform. This is how Tim Berners-Lee, assisted by Robert Cailliau, a Belgian engineer and computer scientist, came up with HyperText Markup Language (HTML), a markup language for representing the structure of a web document using tags added between sentences or words to indicate the role of the text. In order to display it on any terminal regardless of its graphic display capacity[13], HTML is deliberately a very simple language, and above all, without any indication of formatting or possibility of controlling its presentation (excluding any modification of fonts, colors, or text size).

However, as soon as graphical browsers like *Mosaic* (1993) gained popularity, commercial interest in the web brought about an explosion of aesthetic expectations. HTML, initially meant to express semantic structure, was used for visual layout. Designers repurposed elements like `<table>`, transparent images, or even images and Java applets to simulate visual effects[14]. This misuse, although creative, blurred the line between content and presentation and made websites less accessible, maintainable, and semantically meaningful[15]. The influence of print-based graphic design played a role in this confusion. Designers were accustomed to controlling every pixel and page layout aspect, something that the web's model of logical structure followed by visual rendering made difficult to achieve[16].

In 1994, in an attempt to redress the situation and return HTML to its origin as a language for structured documents, Håkon Wium Lie and Bert Bos, both computer scientists, formulated a proposal for Cascading HTML Style Sheets, abbreviated CSS.

With CSS, visual and stylistic rules could be managed independently of the HTML document's structure. This separation became a core architectural requirement of the Web[17]. In practice, it enabled greater adaptability: style sheets could render the same content differently depending on screen size, media type, or user preferences.

The possibilities for page layout have increased rapidly over the last thirty years. Since the introduction of the concept of responsive design in 2010[18], a growing collection of CSS features has emerged that makes it easier to design web pages with adaptive layouts.

> *The layout of a document means (…) the overall graphical structure of its elements when they are displayed on the screen, as opposed to other stylistic information such as fonts or colors. They are not completely separated, of course, because indenting or coloring a text influences what the user perceives as the visual structure of a page. But layout is usually situated at a higher abstraction level than those aforementioned presentational aspects[19].*

Around 2010, two major layout improvements were added to CSS, making it possible to do away with the obsolete technique of HTML tables and float-based positioning[20]. These new tools greatly improved the design and flexibility of responsive web pages. First, Flexbox, a one-dimensional layout model, allows items within a container to expand or contract in order to occupy available space, organizing them along a row or column depending on the container's properties. Second, CSS Grid provides a two-dimensional layout system that structures content into rows and columns, offering many features to simplify the development of complex page designs. A crucial feature shared by these new layout systems is their ability to visually position elements (images, blocks, paragraph, etc.) independently of the source order of the semantic part (i.e., the linear sequence defined by the HTML markup)."

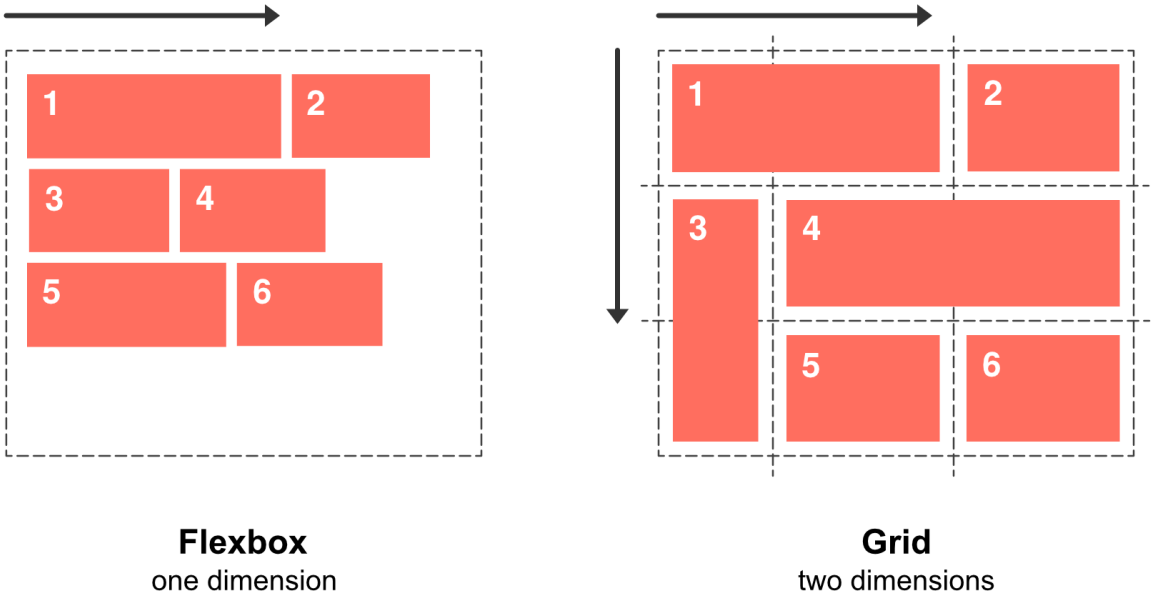

Figure. Flexbox vs. Grid methods in CSS. Numbers show each method's default flow direction.

With CSS Grid, web layout has come full circle, re-integrating principles familiar from traditional graphic design. As noted by Ambrose and Harris[21], layout is about the management of form and space, enabling complex information to be structured in a visually navigable way. CSS Grid echoes these principles, relying on horizontal and vertical axes, proportional areas, and spatial rhythm, just as Swiss-style design grids did in the mid-20th century[22].

César Fernandez-Acebal, who wrote a thesis that originated the CSS grid proposal[23], explicitly draws this connection: Grid in CSS is modeled on the classical theory of layout, enabling a high-level abstraction of visual structure, independent of the logical document order.

Despite this evolution, Masonry layouts — a popular, Pinterest-style staggered grid — remain an unsolved problem in native CSS. This layout requires items of unequal height to be arranged without gaps, in a column-wise flowing pattern. Layout still remains one of the hardest aspects of CSS[24], not only because of its complexity but because it embodies a tension between *specific cases* and *generic thinking* (abstraction). Current discussions on the implementation of Masonry layouts provide us with a framework for exploring the design of the CSS language itself and the question of layout abstraction in the context of the web.

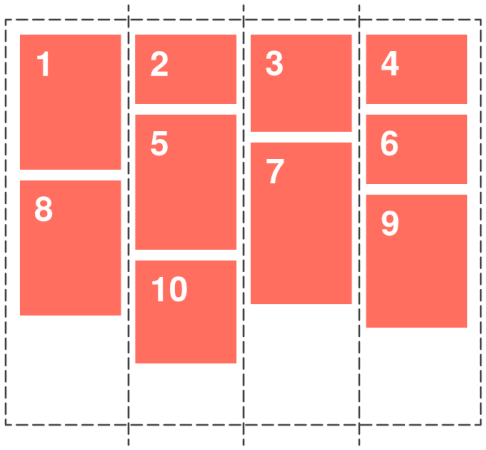

Figure. Masonry layout, with default flow direction of elements.

Investigating Web Standards: Methodological Approach to the W3C Debate

Our research combines a mixed-method approach, combining quantitative and qualitative methods, to trace the evolution of the debate around the Masonry layout feature in CSS.

This new feature was discussed primarily on the public online spaces of the World Wide Web Consortium (W3C), a global organization founded in 1994 by Tim Berners-Lee, the inventor of the Web. The W3C's mission is to ensure the long-term growth of the Web by creating open[25], consensus-based standards.

In information technology, standards enhance compatibility between different programs and hardware, making interoperability essential for resource sharing and collaboration. For example, the web relies on backward compatibility, meaning that nothing should break over time — a website coded twenty years ago should still display correctly today (assuming it hasn't been removed from its server, which is another matter). This principle allows us to still access and enjoy the first web page published in 1991, along with its source code. This longevity is possible because HTML and CSS are standards built around the concept of progressive enhancement through accumulation. Their specifications are published publicly, allowing them to be utilized by anyone in the same way, including web browser manufacturers.

The W3C operates through a structure of working groups, task forces, and community groups. Each working group focuses on a specific aspect of the web. The CSS Working Group (CSSWG) is responsible for developing and maintaining the CSS specifications[26]. Participants in these groups typically include engineers and designers employed by browser vendors (such as Google, Apple, Mozilla, and Microsoft), independent experts, researchers, and other stakeholders from the web community[27].

Since its third version, CSS has been divided into modules, which are families of properties dedicated to a particular domain: text manipulation (CSS Text Module), colors (CSS Color Module), grids (CSS Grid Layout Module), box models (CSS Box Model Module), etc. Each of these modules evolves independently and is assigned a maturity level. They start out as working drafts and evolve towards W3C Recommendation status as they stabilize.

To reconstruct the history of the Masonry layout proposal, we mapped the different public spaces where discussions occurred, both within and outside of formal W3C processes. Internal W3C communication channels include IRC logs (online chat), newsletters, GitHub repositories, and in-person meetings such as TPAC (Technical Plenary and Advisory Committee). In parallel, external spaces such as browser vendors' blogs, personal blogs of CSS authors, social media platforms, and other platforms (CodePen, css-tricks.com, smashingmagazine.com…) illustrate how individual contributors extend and reflect on formal debates.

To systematically study the discussions, we employed a two-phase methodological approach, combining automated data extraction with qualitative analysis. Our main source is GitHub, an online collaborative developer platform, where the W3C systematically shares public documentation, code, and some communication from the organization. CSS specifications are precisely discussed as issues in a dedicated repository (https://github.com/w3c/csswg-drafts/). These issues act as public and searchable threads that document technical debates and community input over time.

Figure. Screenshot of issue #9041 on W3C CSSWG repository on GitHub

Using the GitHub API and custom Bash and Python scripts, we extracted all issues labeled "css-grid-3" (105 issues) and "masonry" (40 issues). Metadata such as issue titles, authors, creation dates, comment counts, and cross-references to other discussions were systematically collected. Our scripts and their results are published publicly on GitLab[28]. Particular attention was paid to the activity of bots logging IRC discussions ("css-meeting-bot", 9 comments), as they often reveal how informal exchanges transition into formal proposals.

However, quantitative analysis rapidly showed its limitations. Metrics such as the number of comments or contributors did not necessarily reflect the conceptual significance of a given discussion. Consequently, a qualitative phase was necessary. We selected relevant issues for close reading based on four main criteria: frequency of citation across other issues, number of comments, density and diversity of engagement, and relevance to the broader conceptual debate around abstraction and layout models. Five issues emerged as central to understanding the debate surrounding the Masonry feature: #945 *[css-grid] [css-flexbox] Pinterest/Masonry style layout support*, #4650 *[css-grid] Masonry layout*, #9041 *Alternative masonry path forward*, #10233 *[css-grid-3] Designer/developer feedback on masonry layout*, #11243 *Alternative masonry path forward*.

Issue number	Title	Date	Opened by	Number of comments	Unique authors
945	[css-grid] [css-flexbox] Pinterest/Masonry style layout support	Jan 16, 2017	rachelandrew	59	30
4650	[css-grid] Masonry layout	Jan 6, 2020	MatsPalmgren	54	22
9041	Alternative masonry path forward	Jul 7, 2023	bfgeek	133	65
10233	[css-grid-3] Designer/developer feedback on masonry layout	Apr 19, 2024	jensimmons	122	95
11243	[css-grid-3] [masonry] Masonry Syntax Debate	Nov 19, 2024	fantasai	28	14

Exploring the Case of Masonry Layouts

The debates around the integration of Masonry layout into CSS offer a rich case study to understand how CSS standards are designed. They reveal how abstraction in CSS is not only a technical or aesthetic issue, but also the outcome of negotiations between different types of arguments: conceptual, technical, interface-centered (CSS author-facing), and political.

Masonry layout refers to a visual organization pattern where items of varying heights are tightly arranged along one axis (typically vertically) without creating gaps, much like stones fitted together in a wall (hence the name). It's also sometimes called "waterfall layout"[29], as a metaphor for how content flows down the page like a waterfall. The origins of the Masonry layout pattern lie outside of CSS. Initially popularized through the JavaScript library *Masonry.js*, developed by David DeSandro, this layout became widespread in web design because it allows content of different sizes to be displayed in a condensed form. Despite this, CSS did not provide a native mechanism for creating Masonry layouts. Developers and designers still rely on JavaScript or complex workaround techniques. Using JavaScript for layout often makes websites slower and creates difficulties for accessibility. This is why a native CSS approach is preferred, as it generally performs layout better, is easier to manage in the long term, and naturally adapts to different screen sizes.

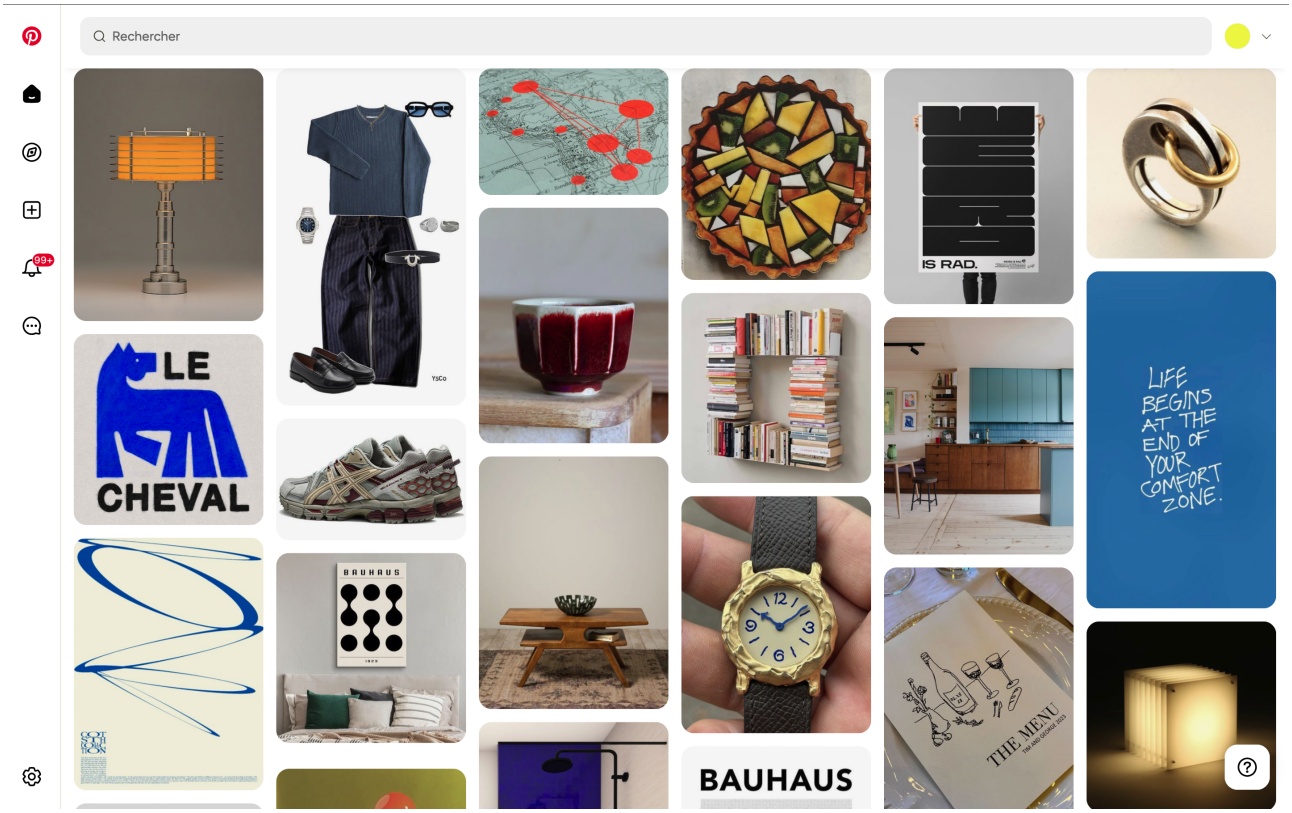

Figure. Screenshot of the Pinterest platform, the most famous Masonry layout on the web.

Origins and Initial Proposals (Grid vs standalone)

Although traces of discussions around Masonry and Pinterest-like layouts can be found in the W3C www-style mailing list as early as 2013[30], the first issue on the GitHub repository of the W3C that mentions this layout is issue #945, opened on January 16, 2017 by Rachel Andrew (@rachelandrew), an independent Web developer and writer at the time. In the first message, Rachel Andrew highlights the web development community's interest in native CSS support for Masonry. The discussions in the issue occur mainly at a conceptual level, questioning whether Masonry should integrate with some layout methods that already exist in CSS (we'll come back to this). Some web developers also present the solutions they were already using.

On January 6, 2020, Mats Palmgren (@MatsPalmgren), layout engineer at Mozilla (Firefox browser vendor), opened a long issue called *#4650 [css-grid] Masonry layout*. He proposed to extend CSS Grid to support Masonry layout "in one of the axes while doing normal grid layout in the other". Quickly, an implementation of this proposal was made available in Firefox Nightly, an experimental version of the Firefox browser that includes the latest features and updates, primarily intended for developers and early testers. Some coded demos were also designed by Jen Simmons graphic designer and CSSWG member (also from the Firefox team at this time), to compare different possible options (Masonry added to Grid, Multicolumn, and Flexbox)[31].

Twenty days later, the CSSWG adopted this Masonry layout proposal and nominated Tab Atkins Jr. (@tabatkins), Elika J. Etemad (@fantasai), and Jen Simmons (@jensimmons) as editors, responsible for tracking issues, responding to feedback, editing the specifications, and driving progress on this specific CSS module [32]. At the end of the year, they transformed the proposal into a W3C Editor's Draft called "CSS Grid Layout Module Level 3 [css-grid-3]"[33], the very early stage of the official design phase of a W3C specification.

This issue #4650 marked the formal entry of Masonry into more technical discussions. However, conceptual discussions continued, tightening up on the question of whether Masonry should be integrated directly into CSS Grid or developed as a standalone layout model with its own properties and logic. Three years later, the discussion was revived with issue #9041, opened on July 7, 2023 by Ian Kilpatrick (@bfgeek), Chromium Blink Engineer at Google, who proposed an alternative Masonry path forward as a standalone layout model.

Analysing discussions across this multiple issues, we observe that the arguments in favor of one or the other of these approaches focused on three main points: conceptual debates, technical implementation concerns, and interface-centered considerations (CSS author-facing).

Conceptual Debates

At a conceptual level, the question was whether Masonry should be considered an extension of other existing CSS layout models (like Multicolumn, Grid, or Flexbox) or whether it represented a fundamentally different model requiring its own abstraction. This touches upon the mental models designers use and the philosophical coherence of CSS's layout systems.

The most prominent debate initially revolved around CSS Grid. Proponents of integration, like Mats Palmgren who initiated the discussion, saw Masonry as a "one-dimensional grid"[34] — leveraging Grid's powerful track definition, alignment, and placement capabilities in one axis while allowing content to stack freely in the other. Furthermore, the ability for items to span multiple columns — a common feature in CSS Grid and in popular JavaScript Masonry libraries[35] — was seen as a crucial requirement. This spanning capability became a significant argument favoring Grid integration. However, integrating Masonry into Grid raised fundamental conceptual issues. Critics highlighted the core difference in logic: Grid establishes its structure *before* sizing tracks based on content, while Masonry needs to know item dimensions *during* placement to find the shortest column[36].

CSS Flexbox was also considered as a potential home for Masonry due to its one-dimensional flow, potentially simplifying the stacking behavior. However, Flexbox's lack of handling for item placement and its inability to span elements across "columns" were problematic and failed to cover essential use cases. Some comments referred to Masonry as a true hybrid between Grid and Flexbox, arguing for a new layout model on its own[37].

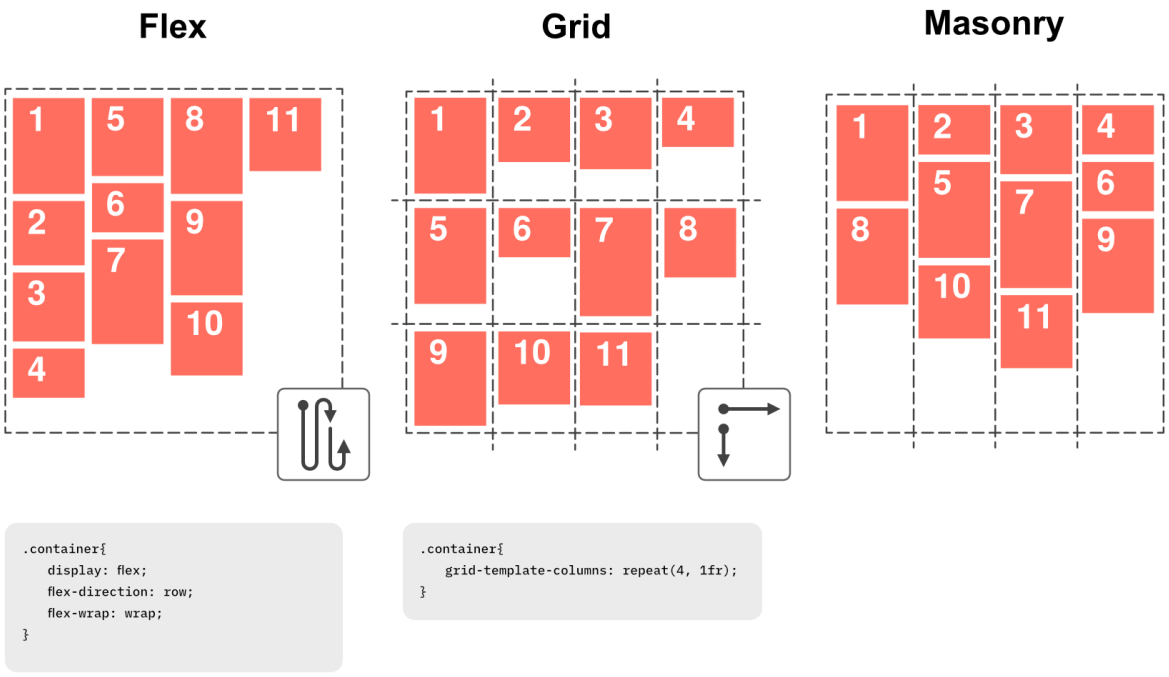

Figure. Comparison of the different CSS layout methods. Numbers show each method's default flow direction.

Uniquely, CSS Multicolumn layout was briefly mentioned as visually similar, distributing content across columns. However, it was quickly dismissed on a conceptual level. As Elika J.Etermad clarified (issue #4650, fantasai on May 6, 2020) , Multicolumn is fundamentally about *fragmentation*; breaking a continuous flow of content into columns, like text in a newspaper. It doesn't involve the item placement logic based on available space that defines Masonry.

This analysis revealed that no existing layout model perfectly accommodated Masonry's unique blend of track-based structure and dynamic stacking, particularly with the requirement for spanning. This conceptual mismatch strengthened the argument that Masonry might require its own layout method.

Technical Implementation Concerns

Beyond the conceptual fit, significant technical hurdles emerged, particularly concerning the performance implications of integrating Masonry into the CSS Grid specification. Engineers from browser vendors like Google (Chrome) and Microsoft (Edge) voiced strong concerns in long comments[38].

Attempts to reconcile this within the Grid specification led to complex algorithms with potentially severe performance drawbacks, described by browser engineers as "quadratic" or even "exponential" complexity. Teams from Chrome and Edge ultimately deemed the integrated approach potentially "unshippable" due to these performance concerns and implementation difficulties, strongly favoring a separate layout method where performance could be better managed by tailoring constraints specifically for Masonry.

In brief, from an implementation standpoint, integrating Masonry into Grid introduces significant code performance and architectural problems.

> *The implementation of Masonry within the Grid Module will require large chunks of divergences in code, which helps indicate to us that Masonry makes more sense as its own display type. (Issue #9041, comment on Apr 24, 2024, by Alison Maher / @alisonmaher)*

Considering the CSS Author Experience

The debate surrounding Masonry's integration into CSS was also significantly influenced by considerations of the author interface (how web developers and designers would use the feature). The central question was whether incorporating Masonry within the existing CSS Grid specification (`display: grid`) or establishing it as a new, separate layout mode (`display: masonry`) would offer a more intuitive, learnable, and effective tool for creators.

This question focused on prioritizing learnability and consistency with existing CSS knowledge.

> *Whether we go with "masonry in grid" or "masonry as separate display type", […] our decision here, which should be guided by what's the best interface for authors. (Issue #9041, comment on Apr 24, 2024, by Elika J. Etemad /@fantasai)*

The arguments in favor of the integration of Masonry into CSS Grid highlighted the benefits of leveraging familiarity and reusing existing concepts. Supporting this argument, it was shown how Masonry layout shares fundamental characteristics with Grid, such as arranging items into tracks (columns or rows). Making it a variant of Grid would allow authors to apply their existing knowledge of Grid properties for tasks like defining column widths (`grid-template-columns`) or setting spacing between items (`gap`).

This reuse was seen as crucial for reducing the learning curve and maintaining conceptual consistency within CSS. Furthermore, it offered access to Grid's powerful features[39]. This argument emphasized the practical advantage of building upon a well-known layout method, potentially making Masonry feel like a natural extension for authors already used to CSS Grid. The goal here is to maintain a cohesive mental model, avoiding the proliferation of different ways to achieve similar layout goals.

Conversely, a significant counter-argument focused on the potential for confusion and increased complexity if Masonry were merged with CSS Grid. Forcing them together under one display property could lead to ambiguity: authors would need to constantly understand which Grid properties apply to Masonry, which behave differently, and which are irrelevant. This could make the combined system harder to teach, learn, and use[40].

In addition, a few comments raised concerns about the long-term health of the specification. Merging distinct layout models might necessitate complex rules and exceptions that could potentially complicate future CSS development with odd inconsistencies[41].

Both sides of the author-interface debate aimed to provide the best possible conceptual clarity for Web developers and designers. But they differed fundamentally on whether exploiting existing features (integration in CSS Grid) or prioritizing conceptual separation and clarity (distinct layout model) would best achieve that goal.

Public Debate and TAG Intervention

We observe a peak in discussion activity in April 2024, following the publication of a blog post[42] on WebKit.org, the web browser engine used by Safari. The article was written by Jen Simmons, who has since been recruited by Apple (the Safari browser vendor). In the post, she advocates for the

integration of Masonry into the existing CSS Grid model and includes demos of various use cases (photos, big menu, newspaper layout, cards for a museum), along with detailed code examples using early and experimental implementations of Masonry in Safari Technology Preview. The article also features a section dedicated to the history of grid systems from a graphic design perspective, including reproductions from various reference books to support her argument. Jen Simmons concludes the article by inviting developers to join the discussion and share their feedback with the CSSWG after trying out the demos. To facilitate this, she opened a specific issue on the W3C repo, named *#10233 Designer/developer feedback on masonry layout*.

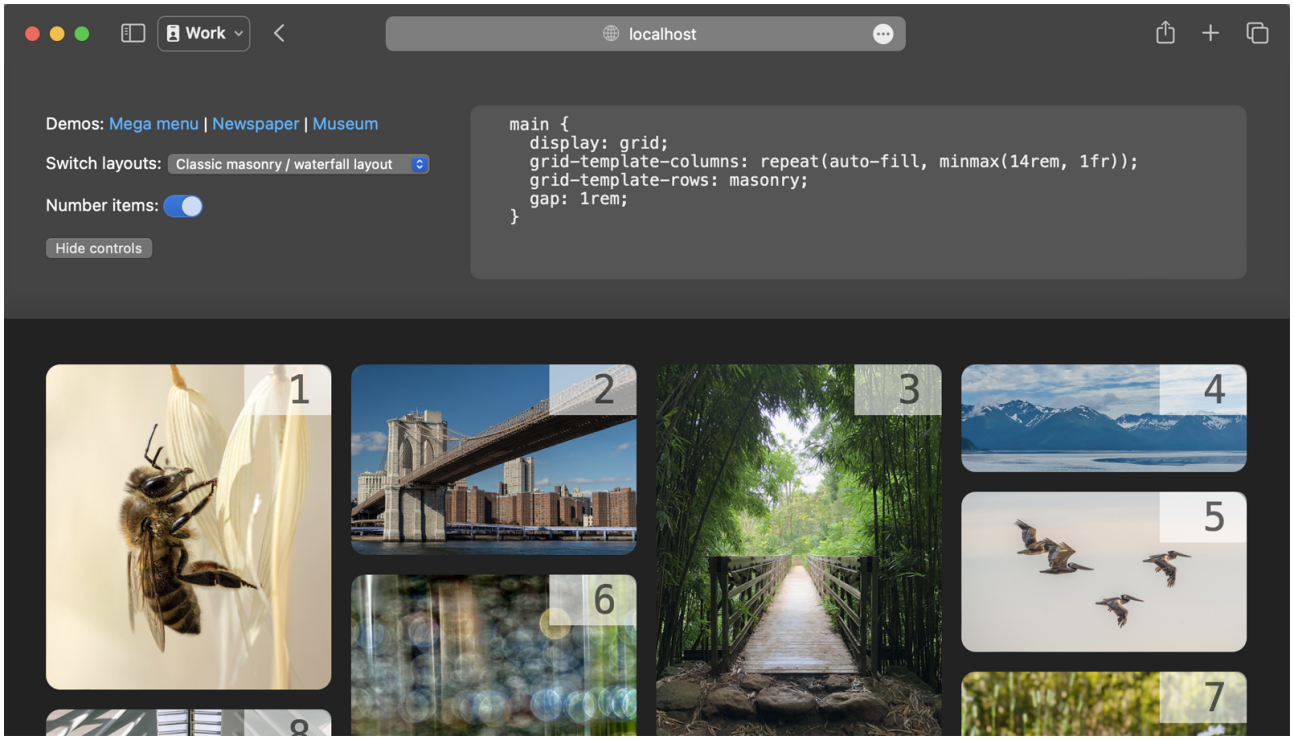

Figure. Demos of Masonry layout in CSS Grid by Jen Simmons.

The same week, engineers from the Chrome and Edge teams published two lengthy comments on issue #9041 to express their position in favor of adopting Masonry as an independent layout method (standalone).

On the same day, on April 24, 2024, a long and detailed discussion in the IRC log posted on the issue #9041 involved several members of the CSSWG (17 members), some of whom had not previously participated in the debate. A preference emerged for integration into CSS Grid, with the reasoning that the proposed syntax better aligns with the expectations of CSS authors[43].

However, despite this, the debate continued for several weeks, and consensus was hard to reach. As a result, the debate went beyond the CSSWG and became more public. In the autumn of 2024, browser vendors and CSS experts began to engage more publicly, publishing blog posts soliciting wider feedback from the web development community[44]. These blog posts played a crucial role in translating technical debates into accessible narratives, inviting designers and developers outside the CSS Working Group to participate.

The impact was immediate and visible on the GitHub discussion threads mentioned in various articles. For example, issue #10233, opened by Jen Simmons following her article, received comments from 95 different contributors. Similarly, issue #9041, cited in Chrome's developer article, saw 65 new comments posted between September 19 and October 5, 2024. Many of these comments were from individuals previously uninvolved, identifying themselves as web developers, CSS designers, accessibility experts, graphic designer and more. During these discussions, many people expressed a preference for a solution that feels like a natural extension of Grid, rather than an entirely new abstraction, while others requested syntax that minimized complexity, even if

sacrificing some flexibility. Expert CSS developers like Ahmad Shadeed[45] and Keith J. Grant[46] also contributed with blog posts, participating in the debate outside the W3C.

As debate intensified, the W3C Technical Architecture Group (TAG) intervened. On October 14, 2024, Elika J. Etemad (Apple) and Tab Atkins-Bittner (Google), both CSS Working Group members, formally requested an early TAG review of the Masonry specification proposals[47], accompanied by an overview of the debate. The TAG's role is to ensure that emerging web features align with principles of architecture and long-term consistency of the web. Their very lengthy feedback, delivered on November 20, 2024, favored a unified set of properties across all layout methods and pushed for deeper unification than either proposal initially offered.

> *Overall, we think Masonry, Grid, and wrapping Flexbox should be incorporated into a unified set of properties. Chrome's [New Masonry Layout] proposal splits apart property sets too eagerly, but even the WebKit [using CSS Grid] proposal seems to miss a chance to develop more-general properties. (…) CSS currently has 3 layout modes (…): Grid, Multicol, and wrapping Flexbox. This is already causing a lot of author confusion, and Masonry attempts to add a 4th mode. As a general principle, having vastly different ways to accomplish slightly different things is a usability antipattern. We urge the [CSS] W[orking] G[group] to explore ways to unify these so that authors can port more knowledge from one to the other (even if they are implemented as separate code paths internally). (Issue #1003 on w3ctag/design-reviews, comment on Nov 20, 2024, by Jeffrey Yasskin /@jyasskin)*

Following the TAG review, discussions within the CSS Working Group shifted, starting to explore ways to harmonize Masonry behaviors with existing layout controls, unifying `grid-auto-flow` (CSS Grid) and `flex-flow` (CSS Flexbox) properties[48]. This whole idea for a unifying "Item Flow" layout method is still a work in progress. At the time of writing, the WebKit team from Apple has just published a blog post explaining their reflection on this new concept[49].

This process clearly illustrates the socio-political design of CSS standards. Throughout this evolution, the Masonry debate highlights how spaces of standardization are not monolithic and are constantly evolving. GitHub issues hosted technical exchanges, blog posts and online demonstrations linked formal specifications to community conversations, and formal TAG reviews focused discussions on architectural principles. Together, these spaces illustrate a distributed discussion where design abstractions are collectively negotiated.

Conclusion

Our exploration of the Masonry layout debates has illustred the complex processes supporting the design and evolution of CSS. By examining this specific case, we have aimed to understand CSS not merely as a tool *for* web design, but as a *designed object* in its own right — a formal system shaped by historical contingencies, technical constraints, and collective negotiation with considerations for user and sustainability.

The journey of Masonry highlights that standardization within the W3C is far from a linear or purely technical process. Rather, it manifests as a dynamic, multi-sited negotiation where technical reasoning, conceptual modeling, practical authoring experience, and governance strategies intersect. This process reveals the intricate ways in which a diverse range of actors — engineers, browser vendors, standardization experts, and the wider community of graphic designers and web developers — contribute to the ongoing construction of CSS.

Debates often arise about whether standards should primarily originate from implementations, or be driven by designers' needs, or be crafted by the standards body itself. However, examining the Mansonry debate reveals that valuable contributions have emerged from all these sources[50]. These

contributions, often formulated in highly specialized technical terms, reflect the underlying challenge: **collectively developing and appropriating shared representations of graphic design concepts within the formal constraints of code**.

At the heart of this challenge are the concept of abstraction, which operates at two distinct levels in our exploration. Firstly, designers *using* CSS engage in abstraction daily. They translate visual layout into declarative rules, creating formal systems that must anticipate a variety of display contexts and define the relational behavior of elements, rather than fixing their appearance into a pixel-perfect way. Secondly, the very *creation* of CSS involves a profound act of abstraction. Crafting the language itself — defining its rules and properties — requires anticipating potential use cases and generalizing visual patterns into reusable, formal descriptions.

On a philosophical point of view, If we understand abstraction, in line with its Latin root *abs-trahere* (to separate the essential from the detail), as closely related to generalization, then **CSS specifications can be seen as practical instantiations of theoretical ideas, generalized to serve a broad community**. Abstraction becomes the method for articulating the *essence* of layout problems, delegating the specifics to the browser's rendering logic. Ultimatly, abstraction is a tool for design.

Finally, the case of CSS Masonry underscores how intimately the work of web design is entangled with its material medium — the code itself. Understanding CSS as both an abstraction system and a collaboratively constructed artifact reveals the socio-technical dynamics that shape our digital tools. This perspective highlights the importance of designers actively participating in building their own tools, which constitutes a form of meta-level collaboration[51]. The ongoing evolution of CSS demonstrates this potential, indicating that the design community can influence its own technical culture and practices through such engagement.

References

Ambrose, Gavin, and Paul Harris. *Grids*. Lausanne: AVA Academia, 2008.

Anderson, Richard J., and Sumeet Sobti. 'The Table Layout Problem'. In *Proceedings of the Fifteenth Annual Symposium on Computational Geometry*, SCG '99, New York, NY, USA: Association for Computing Machinery, 1999, pp. 115–23. https://doi.org/10.1145/304893.304937.

Badros, Greg J., Alan Borning, Kim Marriott, and Peter Stuckey. 'Constraint Cascading Style Sheets for the Web'. In *Proceedings of the 12th Annual ACM Symposium on User Interface Software and Technology*, UIST '99, New York, NY, USA: Association for Computing Machinery, 1999, pp. 73–82. https://doi.org/10.1145/320719.322588.

Berners-Lee, Tim. *Weaving the Web: The Original Design and Ultimate Destiny of the World Wide Web*. 1st ed. San Francisco: Harper Business, 2000.

Bos, Bert. 'A Brief History of CSS until 2016'. w3.org, 17 December 2016. https://www.w3.org/Style/CSS20/history.html.

Butler, Judith. *Excitable Speech: A Politics of the Performative*. New York: Routledge, 1997.

Détienne, Françoise. *Software Design : Cognitive Aspects*. Springer Verlag, 2001. https://hal.inria.fr/inria-00117292.

Farge, Odile. 'La rhétorique de la conception. Pour une conscientisation du rôle de l'outil dans la formation d'une culture numérique'. *Interfaces numériques* 4, no 3 (December 2017): 540–540. https://doi.org/10.25965/interfaces-numeriques.481.

Fernández Acebal, César. *ALMcss: Separación de estructura y presentación en la web mediante posicionamiento avanzado en CSS*. PhD thesis, Universidad de Oviedo, 2010.

https://digibuo.uniovi.es/dspace/handle/10651/12715.

Grant, Keith J. ‘Resilient, Declarative, Contextual’. *Https://Keithjgrant.Com/* (blog), June 2018. https://keithjgrant.com/posts/2018/06/resilient-declarative-contextual/.

Hayles, N. Katherine. *My Mother Was a Computer: Digital Subjects and Literary Texts*. Chicago: University of Chicago Press, 2005.

Korpela, J. ‘Lurching toward Babel: HTML, CSS and XML’. *Computer* 31, no 7 (July 1998): 103–4. https://doi.org/10.1109/2.689682.

Kramer, Jeff. ‘Is Abstraction the Key to Computing?’ *Commun. ACM* 50, no 4 (April 2007): 36–42. https://doi.org/10.1145/1232743.1232745.

Levering, Ryan, and Michal Cutler. ‘The Portrait of a Common HTML Web Page’. In *Proceedings of the 2006 ACM Symposium on Document Engineering*, DocEng ’06, New York: Association for Computing Machinery, 2006, pp. 198–204. https://doi.org/10.1145/1166160.1166213.

Lie, Håkon, and Bert Bos. *Cascading Style Sheets: Designing for the Web, Third Edition*. Addison-Wesley Professional, 2005.

Lie, Håkon Wium. ‘Cascading Style Sheets’. PhD thesis, University of Oslo, 2005. https://www.wiumlie.no/2006/phd/.

Lie, Häkon Wium. ‘CSS and User-Adapted Web Presentations’. In *Proceedings of the 2017 Conference on Conference Human Information Interaction and Retrieval*, 5, CHIIR ’17, New York, NY, USA: Association for Computing Machinery, 2017. https://doi.org/10.1145/3020165.3038294.

Loanardi, Paul M. ‘Digital Materiality ? How Artifacts without Matter, Matter’. *First Monday* 15, no. 6 (2010).

MacKenzie, Adrian. *Cutting Code. Software and Sociality*. Science, Society & Culture. Peter Lang, 2006.

Mackenzie, Adrian. ‘The Performativity of Code: Software and Cultures of Circulation’. *Theory, Culture & Society* 22, no. 1 (1 February 2005): 71–92. https://doi.org/10.1177/0263276405048436.

Maurer, Luna, Edo Paulus, Jonathan Puckey, and Roel Wouters, eds. *Conditional Design Workbook*. Amsterdam: Valiz, 2013.

McCullough, Malcolm. *Abstracting Craft: The Practiced Digital Hand*. Cambridge, Mass.: MIT Press, 1998.

Müller-Brockmann, Josef. *Grid Systems in Graphic Design: A Visual Communication Manual for Graphic Designers, Typographers and Three Dimensional Designers*. Sulgen, Suisse: Verlag Niggli, 1981.

Nakar, Liat, and Michal Armoni. ‘Aiming Towards Abstraction: Does Algorithmic-Pattern-Oriented Instruction Promote the Teaching of Abstraction?’ In *Proceedings of the 56th ACM Technical Symposium on Computer Science Education V. 1*, 812–18. SIGCSETS 2025. New York, NY, USA: Association for Computing Machinery, 2025. https://doi.org/10.1145/3641554.3701914.

1. Kandy, A. J. ‘A DevTools for Designers’. Medium. *UX Collective* (blog), 4 October 2018. https://uxdesign.cc/a-devtools-for-designers-2342aab88c06.
2. Kramer, Jeff, ‘Is Abstraction the Key to Computing?’ *Commun. ACM* 50, no. 4, April 2007, pp. 36–42.
3. We deliberately use the term *designer* in this paper instead of the more common terms *developer* or *programmer* in this context, as the programming practices we are focusing on here are primarily centered around styling with CSS and graphic design.
4. Nakar, Liat, and Michal Armoni, ‘Aiming Towards Abstraction: Does Algorithmic-Pattern-Oriented Instruction Promote the Teaching of Abstraction?’ In *Proceedings of the 56th ACM Technical Symposium on Computer Science Education V. 1*, 812–18. SIGCSETS 2025. New York, NY, USA: Association for Computing Machinery, 2025. Emphasis added.
5. In metal type, the point size of a font describes the height of the metal body on which that font’s characters were cast. This unit was kept with the photocomposition process.
6. Paul Brainerd, co-founder of Aldus Corporation and a former graphic designer in photocomposing, explains that most of PageMaker’s interface [ancestor of desktop publishing software] comes from his experience of collaging with a razor blade or scalpel. See Briar Levit’s documentary, ‘Graphic Means: A History of Graphic Design Production’ (2017).
7. Lie, Håkon Wium, ‘Cascading Style Sheets’, PhD thesis, University of Oslo, 2005. See online: https://www.wiumlie.no/2006/phd/.
8. CSS includes far more capabilities than simply supporting diverse screen sizes. It addresses a wide range of design needs, such as improving accessibility, enhancing maintainability, ensuring device and platform independence, allowing multiple style sheets for the same document, forward and backward compatibility, and providing flexible rendering across different media types like print, braille, and speech. However, since layout is the main concern in this discussion, we will primarily focus on the layout features of CSS. For more on CSS capabilities, see the W3C CSS Design Principles: https://www.w3.org/TR/CSS22/intro.html#design-principles.
9. “Conditional design is a design method formulated by the graphic designers Luca Maurer, Jonathan Puckey, Roel Wouters and the artist Edo Paulus, in which conditions and rules of play are drawn up that invite cooperation within a ‘regulated’ process towards an unpredictable design or result.” https://conditionaldesign.org/manifesto/
10. Grant, Keith J, ‘Resilient, Declarative, Contextual’, *Https://Keithjgrant.Com/* (blog), June 2018. https://keithjgrant.com/posts/2018/06/resilient-declarative-contextual/.
11. Imperative languages are usually *general purpose* programming languages, meaning they can be used to program pretty much anything for a wide variety of platforms. Declarative languages, on the other hand, are most often *domain-specific* languages, or DSLs, meaning they were developed to be used for a specific purpose, within a specific domain. CSS is a domain-specific declarative language.
12. See Berners-Lee, Tim, *Weaving the Web: The Original Design and Ultimate Destiny of the World Wide Web*, 1st ed, San Francisco, Harper Business, 2000. On the first page of the Web, we can read: “The WorldWideWeb (W3) is a wide-area hypermedia information retrieval initiative aiming to give universal access to a large universe of documents” http://info.cern.ch/hypertext/WWW/TheProject.html
13. The Web also had to work on machines without graphic interfaces, which were still very common at the time.
14. Levering, Ryan, and Michal Cutler, “The Portrait of a Common HTML Web Page”, in *Proceedings of the 2006 ACM Symposium on Document Engineering*, DocEng ’06, New

York, NY, USA, Association for Computing Machinery, 2006, pp. 198–204.

15. Anderson, Richard J., and Sumeet Sobti, “The Table Layout Problem”, in *Proceedings of the Fifteenth Annual Symposium on Computational Geometry*, SCG ’99, New York, NY, USA, Association for Computing Machinery, 1999, pp. 115–23.
16. Korpela, J, “Lurching toward Babel: HTML, CSS and XML”, *Computer* 31, no. 7, July 1998, pp. 103–4.
17. Badros, Greg J., Alan Borning, Kim Marriott, and Peter Stuckey, “Constraint Cascading Style Sheets for the Web”, in *Proceedings of the 12th Annual ACM Symposium on User Interface Software and Technology*, UIST ’99, New York, NY, USA, Association for Computing Machinery, 1999, pp. 73–82.
18. Responsive web design is an approach to web design that ensures websites display effectively across a wide range of devices, screen sizes, and window dimensions. It adapts the layout of a web page to the user’s viewing environment by employing techniques such as fluid layouts, proportion-based grids, flexible images, and CSS media queries. Ethan Marcotte invented the term *responsive web design* in a May 2010 article in *A List Apart*: https://alistapart.com/article/responsive-web-design/
19. Fernández Acebal, César, *ALMcss: Separación de estructura y presentación en la web mediante posicionamiento avanzado en CSS*, PhD thesis, Universidad de Oviedo, 2010, p. 9.
20. Originally intended for wrapping text around images, the `float` property eventually became a popular method for creating multi-column layouts. Aside from absolute positioning, it was one of the few CSS mechanisms that allowed elements to be visually positioned differently from their order in the HTML markup.
21. Ambrose, Gavin, and Paul Harris. *Grids*, Lausanne, AVA Academia, 2008.
22. Müller-Brockmann, Josef, *Grid systems in graphic design: a visual communication manual for graphic designers, typographers and three dimensional designers*, Sulgen, Suisse, Verlag Niggli, 1981.
23. “This thesis proposes a new layout mechanism for CSS, which has been developed within the W3C Cascading Style Sheets Working Group (CSS-WG), co-authored by this author and one of his supervisors [Bert Bos]: the CSS3 Template Layout Module.” Fernández Acebal, César, *op. cit*.
24. Elika J. Etemad, “Evolution of CSS Layout: 1990s to the Future”, *A touch of Class* [online], 10/04/2012, https://fantasai.inkedblade.net/weblog/2012/css-layout-evolution/.
25. The consortium was created to prevent technological fragmentation of the Web by proposing a space for deliberation between members that could lead to a consensus on shared standards, while preventing any single vendor from monopolising the moral and legal ownership of these standards. See Tim Berners-Lee, Mark Fischetti, and Michael L Dertouzos, *Weaving the Web: The Original Design and Ultimate Destiny of the World Wide Web by Its Inventor* (New York: HarperCollins, 2008) , pp. 98-99.
26. To know more about how CSSWG works, Elika J. Etemad’s series of blog posts offers an insightful overview with key aspects such as the people and roles involved, communication methods, decision-making processes, modularization of specifications, and the overall specification development workflow. Elika J. Etemad, “about:csswg, An Inside View of the CSS Working Group at W3C”, *A Touch of Class* [online], 2011, https://fantasai.inkedblade.net/weblog/2011/inside-csswg/
27. CSS Working Group currently has 189 participants (including 17 invited experts) representing 35 organizations, see https://www.w3.org/groups/wg/css/participants/.
28. https://gitlab.com/JulieBlanc/data-from-github-issues
29. https://github.com/topics/waterfall-layout

30. Tab Atkins Jr. “Re: [css-grid] Dense Packing (was: [CSSWG] Minutes Telecon 2013-07-19)”, *www-style* mailing-list, 19 Jul 2013, https://lists.w3.org/Archives/Public/www-style/2013Jul/0486.html
31. https://codepen.io/jensimmons/full/vYNeRZw
32. See the css-meeting-bot log on January 23, 2020: https://github.com/w3c/csswg-drafts/issues/4650#issuecomment-577614598
33. “CSS Grid Layout Module Level 3, Editor’s Draft, 22 October 2020”, https://web.archive.org/web/20201028164253/https://drafts.csswg.org/css-grid-3/
34. “The proposal here is to define a ‘one-dimensional grid’, so that you have tracks in just one axis and a continuous flow (stacking blocks one after another) in the other. So indeed, there are no”shared row lines” anywhere in the masonry axis (except at the start edge perhaps). It seems to me this is precisely what masonry layout is about, one axis has grid-like properties (tracks), while the other axis has a continuous flow (in each track separately).’” (Issue #4650, comment on Jan 22, 2020, by Mats Palmgren /@MatsPalmgren)
35. Desandro’s original masonry library is described as a “Cascading *grid layout* library.’”
36. “At their most fundamental level, Grid and Masonry are opposite with respect to sizing and placement. Grid places all items before layout, and then has complete knowledge of what items are in any given track, so it can do complex intrinsic sizing based on that knowledge. Masonry places items as they’re laid out, and thus it cannot know what elements will end up in any given track, and can’t do the same complex intrinsic sizing.’” (Issue #9042, comment on Apr 24, 2024, by Tab Atkins Jr. /@tabatkins)
37. “To me, grid creates rows and columns (…) where Flexbox allows for more intrinsic layouts that have no lines, but does have an axis. Masonry is a combo of both in most cases, where columns are desired (so vertical lines) but no horizontal lines, as all items can have their own intrinsic height. (…) Masonry shares more with Flexbox than Grid in my opinion.” (Issue #4650, comment on Jan 21, 2020, by Adam Argyle /@argyleink)
38. See the comments on issue #9041, from Tab Atkins Jr. (Google) and Alison Maher (Microsoft Edge) posted both on Apr 24, 2024.
39. “I believe Masonry-style layout belongs in Grid. (…) Making this part of Grid also gives authors all the other powers of Grid — track sizing, names, etc.” (Issue #4650, comment on Jan 23, 2020, by Jen Simmons /@jensimmons)
40. “It seems like we create a lot of additional complexity by making grid do a non-grid thing. Add to that the teaching issue, it’s been tricky enough to explain one-dimensional vs. two-dimensional to authors, and encourage understanding of which layout method to use for which use case. I think that tying Masonry, which is more like Flexbox than Grid, to Grid layout would be ultimately very confusing.” (Issue #4650, comment on Jan 22, 2020, by Rqchel Andrew /@rachelandrew)
41. “I believe this situation to be similar to that of Block and Multicol—these were folded into a single layout mode, and ever since we’ve had to deal with odd inconsistencies between the two in what behaviors they expect. If we had defined display: multicol back in the day, many issues would have been avoided. I think the Grid/Masonry marriage is even more fraught with inconsistencies” (https://github.com/w3c/csswg-drafts/issues/9041#issuecomment-2075210820)
42. Jen Simmons, “Help us invent CSS Grid Level 3, aka ‘Masonry’ layout”, *WebKit.org* [online], April 19, 2024, https://webkit.org/blog/15269/help-us-invent-masonry-layouts-for-css-grid-level-3/
43. One extract of the discussion: “Lea [Verou]: These demos are impressive, and this is solving real author pain points. (..) I did have some reservations about how this combines with multicol from an author point of view, but I think I’m now convinced this makes sense as a part of grid. (…) Miriam: Agree quite a bit with Lea. Agree we like this as part of grid. Syntax feels right. (…) Looking at separate masonry proposal, seems like new terms for

similar things. Why do I need to learn new terms for the same thing? I also might want to switch between grid and masonry at different break points. Keeping them together makes it a lot easier to do”

44. Google Chrome Team (Rachel Andrew, Ian Kilpatrick, Tab Atkins-Bittner), “Feedback needed: How should we define CSS Masonry?,” September 19, 2024, https://developer.chrome.com/blog/masonry-syntax; WebKit Team (Jen Simmons and Elika Etemad), “Help us choose the final syntax for Masonry in CSS,” October 21, 2024, https://webkit.org/blog/16026/css-masonry-syntax/.
45. “Should masonry be part of CSS grid?”, Oct 30, 2024, https://ishadeed.com/article/css-grid-masonry/
46. “Weighing in on CSS Masonry”, May 2024, https://keithjgrant.com/posts/2024/05/weighing-in-on-css-masonry/
47. The issue *#1003 CSS Masonry Layout* was opened on the GitHub TAG group by Elika J. Etemad / @fantasai, https://github.com/w3ctag/design-reviews/issues/1003
48. https://github.com/w3c/csswg-drafts/issues/11480
49. Jen Simmons and Elika Etemad, “Introducing Item Flow: a new layout mode encompassing grid, flexbox, and masonry,” *WebKit.org* [online], December 16, 2024, https://webkit.org/blog/16082/introducing-item-flow-a-new-layout-mode-encompassing-grid-flexbox-and-masonry/
50. This observation aligns with Elika J. Etemad’s work, which demonstrates a similar point for other CSS properties. See “about:csswg, An Inside View of the CSS Working Group at W3C”, *op. cit.*
51. We have developed this idea in another publication: Julie Blanc, “Large-scale collaboration in graphic design communities of practice”, *Our Collaborative tools*, [online], 2024, https://ourcollaborative.tools/en/article/collaboration-in-graphic-design-commun