



Contribution & Confidence

Rachel Andrew | All Things Open 2016

Hello

How would I get started
today?

It's 1989



Rachel Andrew | Tweet @rachelandrew #ato2016



Dancers do not need to use
computers



The web gave me a
community

Welcome



For proper placement of text and graphics...
Please feel free to check out my document "Source Code"

Please do not alter my graphics in any way.

It would be appreciated if you would let me know if you do use any of my graphics...

I would love to pay your site a visit to see what you have done.



Sign



*Guest
Book*



View



To return to the previous page...Just click on the "BACK" Button

To return to My Main Index Page...click on the "HOME" Button



Back



Home



Next



E Mail



Links

“Knowing HTML” was a
marketable skill

Learning something one
day

Teaching it to someone else the next

The web gave me a new
career.

The web was accessible, and
had a culture of sharing
knowledge.

Font tags and nested tables

glish.com : CSS layout tech

glish.com/css/

AppsShow Shapes


GET CLOUD WITH NOTHING TO HIDE.

Bandwidth included.

Storage included.

Total access, control, and transparency, included.

LEARN MORE >



CSS Layout Techniques: for Fun and Profit

Look Ma, No Tables.

If you are looking for help making the transition to CSS layout (that's Cascading Style Sheets), you've come to the right place. I am cataloging here as many useful *cross-browser* CSS layout techniques as I can find, and some that I made up when I was bored last Thursday. All the **examples on this site** have been reduced to only their essential code, and you will find the source displayed on each page to hopefully make it quick and easy to understand the inner workings of the CSS. **Feel free to steal all the code you find on this site**, and consider linking back here on your site or in your source comments.

You will also find below links to various online CSS **resources** and **tutorials**, appropriate for both the novice and the seasoned CSS veteran.

I started this collection because of the dearth of resources I found out there when I went looking for information on how to translate typical table based layouts to CSS layouts. I know it is not nearly exhaustive, so if you see that there is something missing, whether it is a particularly good tutorial, or a site that is using a complex CSS layout, please **let me know about it**. I will pay you \$3750 for each link you submit that I use.

If you don't have any idea why anyone cares about this topic, because like tables can do all that stuff and more, please read this: **To Hell with Bad Browsers**. And then read this **follow up interview** with Zeldman. And then read about the Web Standards Project's **Browser Upgrade** campaign. The future is bright, kids!

CSS Techniques

CSS layout techniques and the sites that use them.

3 Columns, The Holy Grail

of page layouts — The most elegant technique and perhaps the most sought after layout: a 3 column page with a fluid center column. Easy to understand, easy to implement. I first saw this layout at **dynamic ribbon device** and have since learned that the sweet CSS came from Rob Chandanaia of **BlueRobot**. Owen also made a very nice **tutorial** using this layout technique.

2 Columns, ALA Style

— Famously chronicled by Jeffrey Zeldman in his ALA article **A Web Designer's Journey**, this is an extremely easy layout to implement requiring only a simple float:left declaration.

4 Columns, All Fluid

— This technique can actually be used to provide as many columns on a page as you like. Drawback #1) it gets difficult quickly if you want to make any of the columns a fixed width. Drawback #2) it relies heavily on percentages, which the various browsers all calculate differently, so you can't place your columns very precisely. Still, a very useful technique, especially if you don't want borders and different background colors for your columns.

3 Columns, All Fluid

— A much simpler and potentially more useful technique than 4 column technique above. It uses float:left, suffers from needing percentage widths for each column, and from potential column wrapping when the browser window is narrowed.

Static Width and Centered

— 3 columns all with static widths, and contained in a parent DIV which remains centered in the window. One rather serious limitation of this particular technique is that if any of the three content DIVs contains an image or a really long word that is longer than the width of the DIV, it totally breaks the layout. Each browser breaks it differently.

Nested Float

— A very simple layout that features a nested, floated menu in the upper right. Easily reversed. A variation of this technique is in use on this very page.

CSS Resources

Specs, primers, validators, stuff like that

The CSShark Answers FAQs

— Martina Kosloff has compiled a pretty good FAQ on CSS. Worth your time.

css/edge

— From the mind of Eric Meyer comes this great little site pushing CSS to the edge. It is, in his words: "intended, first and foremost, to be as relentlessly creative with CSS as we have been practical all these years. It does **not** exist to present or explain safe cross-browser techniques; in fact, almost the opposite. The goal here is to find ways to make CSS live up to its fullest potential, with only minimal regard to browser limitations."

websitetips.com CSS section

— Literally a ton of links to CSS resources from all over. A better set of links than this one by far.

Guide to Cascading Style Sheets

from the Web Design Group. — An excellent primer if you need to start from scratch. It features a tutorial, a reference section, a syntax validator, and more CSS links. A little dated, but still an excellent place to start.

Style Sheet Reference Guide

from webreview.com. — Eric Meyer's excellent resource, including the justly famous browser compatibility charts. It's a bit dated (I wish it included Opera 5 and more of CSS2) but still an excellent resource.

CSS School

from w3schools.com. — A whole lot of information, references, and examples.

Quick-Links to the Layouts:

3 columns, the holy grail

2 columns, ALA style

4 columns, all fluid

3 columns, all fluid

static width and centered

nested float

On This Page:

CSS Techniques

CSS Resources

CSS Tutorials

At This Site:


Blogger Template

CSS Hacks Explanation

Celebrity Makover

What's New:

5-04-04 — The second edition of our book is out:



5-29-02

— The book is out: **Cascading Style Sheets: Separating Content from Presentation**.

9-1-01

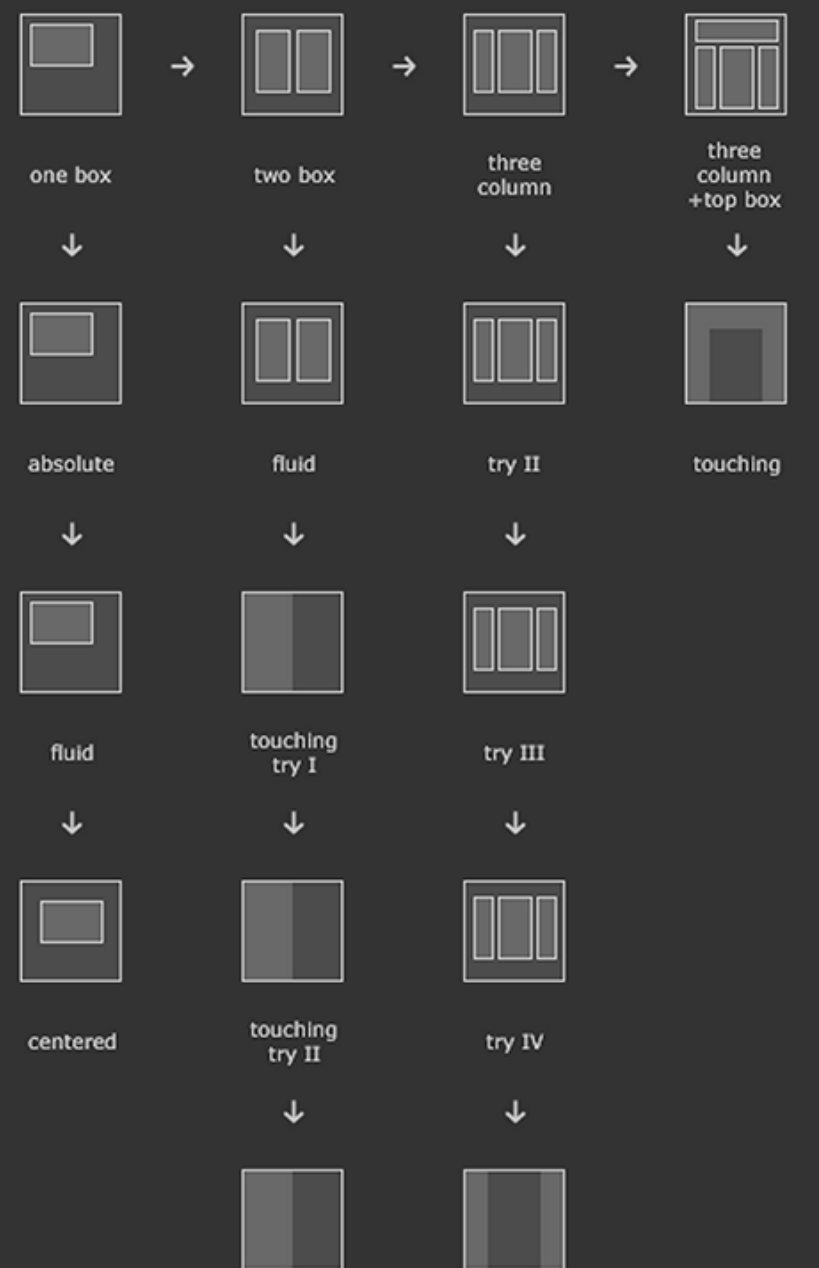
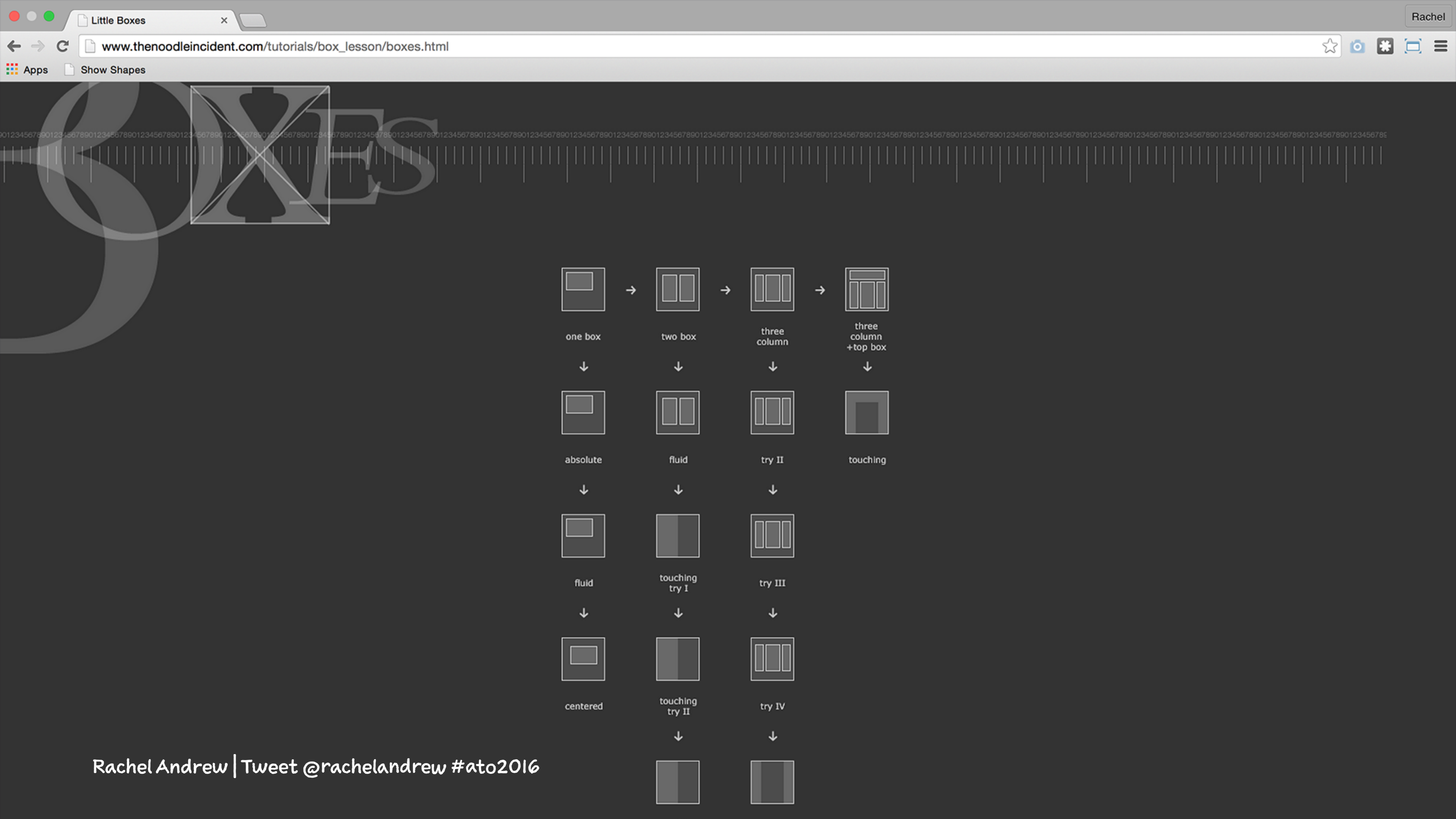
— **I'm looking for a job**. Know anybody in NY city that needs a web developer with XML/XSLT, ASP, SQL, DHTML, and CSS skills? **Let me know!**

9-1-01

— Happy September. I just added four new links to outside resources (see below), all of them great: **The CSShark Answers FAQs**, **First Steps in Cascading Style Sheets**, **200**

straight to the top

Rache





edge of my seat . com

Web Design and Development Services

[home](#)[services](#)[free stuff](#)[about](#)[about us](#)
[contact](#)

All about edgeofmyseat.com

We are a small web design and development company located in Berkshire, United Kingdom although we will work globally.

We specialise in creating dynamic, data-driven websites for companies and non-profit organisations that allow them to use the web - not as a brochure - but as an alive medium of community and information that can extend the reach of their operations.

We build complete solutions, working with our clients to create a look and feel that compliments their existing branding while providing a comfortable user experience.

We will also work with other agencies and designers, as Development Partners, to create dynamic data-driven sections of existing sites or work in progress.

Find out more about [our services](#) or [contact us](#) to find out how we can help you.

Upgrade to a better Web experience.

WEB BUILDERS:

Tired of hacks and versioning? Write **valid** markup and code, optimizing it to degrade reasonably on older, non-compliant browsers. If you must, send non-compliant browsers to this page, or (better still) to your own Browser Upgrades page, crafted to the needs of your particular audience. If your site relies on the W3C DOM or CSS layouts, an upgrade page could be of great service to your readers. Our **Tips**

Page can assist

BROWSER UPGRADES

How did I get here?

The folks who built the site you were trying to visit (">) have directed you to this page because your browser does not support accepted web standards. (Or you may have simply followed a link to this page.)

What “web standards?”

The ones created by the **World Wide Web Consortium** (W3C) — the people who invented the web itself. The W3C created these standards so the web would work better for everyone. New browsers, mainly, support these W3C standards; old browsers, mainly, don’t.

What can I do?

You might consider upgrading to any of the following browsers. Doing so will allow you to use and view websites as their creators intended.

IE6 for Windows delivers fine support for HTML 4, CSS-1, and other important W3C standards. Don’t worry if you don’t know what that means; the people who build your websites know. The browser is available free of charge.

IE5 Macintosh Edition, released in March 2000, provides superb support for key web standards (CSS, HTML, XHTML, PNG, ECMA-262, DOM1HTML) and an elegant user experience. IE5.1, released December 2001, improves on its predecessor. The browser is available free of charge.

Netscape 6.2 complies with important Web standards, including

Encouraging designers to
care about web standards



Front-end developer 2005?



Browser bugs expert

Thanks to the hard work of countless
WaSP members and supporters (like
you), Tim Berners-Lee's vision of the
web as an open, accessible, and
universal community is largely the
reality.



<http://www.webstandards.org/2013/03/01/our-work-here-is-done/>

Browser vendors are
implementing standard
things in a standard way

Innovation happens
through the standards
process

1	Introduction
2	Overview
2.1	Declaring the Grid
2.2	Placing Items
2.3	Sizing the Grid
2.4	Background and Motivation
2.4.1	Adapting Layouts to Available Space
2.4.2	Source-Order Independence
3	Grid Layout Concepts and Terminology
3.1	Grid Lines
3.2	Grid Tracks and Cells
3.3	Grid Areas
4	Reordering and Accessibility
5	Grid Containers
5.1	Establishing Grid Containers: the 'grid', 'inline-grid', and 'subgrid' 'display' values
5.2	Sizing Grid Containers
5.3	Clamping Overly Large Grids
6	Grid Items
6.1	Grid Item Display
6.2	Grid Item Sizing
6.3	Reordered Grid Items: the 'order' property
6.4	Grid Item Margins and Paddings
6.5	Z-axis Ordering: the 'z-index' property
6.6	Implied Minimum Size of Grid Items
7	Defining the Grid
7.1	The Explicit Grid
7.2	Explicit Track Sizing: the

CSS Grid Layout Module Level 1

W3C Candidate Recommendation, 29
September 2016

This version:

<https://www.w3.org/TR/2016/CR-css-grid-1-20160929/>

Latest published version:

<https://www.w3.org/TR/css-grid-1/>

Editor's Draft:

<https://drafts.csswg.org/css-grid/>

Previous Versions:

<https://www.w3.org/TR/2016/WD-css-grid-1-20160519/>

<https://www.w3.org/TR/2015/WD-css-grid-1-20150917/>

<https://www.w3.org/TR/2015/WD-css-grid-1-20150806/>

<https://www.w3.org/TR/2015/WD-css-grid-1-20150317/>

<https://www.w3.org/TR/2014/WD-css-grid-1-20140513/>

<https://www.w3.org/TR/2014/WD-css-grid-1-20140123/>

<https://www.w3.org/TR/2013/WD-css3-grid-layout-20130402/>

<https://www.w3.org/TR/2012/WD-css3-grid-layout-20121106/>

Feedback:

www-style@w3.org with subject line "[css-grid] ... message topic ..." ([archives](#))

Test Suite:

http://test.csswg.org/suites/css-grid-1_dev/nightly-unstable/

Issue Tracking:

[Disposition of Comments](#)

[Inline In Spec](#)

Editors:

[Tab Atkins Jr.](#) (Google)

[Elika J. Etemad / fantasai](#) (Invited Expert)

[Rossen Atanasov](#) (Microsoft)

Former Editors:

[Alex Mogilevsky](#) (Microsoft Corporation)



Show stopping browser
bugs when doing
straightforward things in
modern browsers are rare

The industry has grown up

Studies show that a todo list is the most complex JavaScript app you can create before a newer, better framework is invented.



<http://www.allenpike.com/2015/javascript-framework-fatigue/>

We're creating complexity

Hiding the simple languages of the web
behind tooling and process



Rachel Andrew
@rachelandrew

 Follow

It may be due to my being old, but as I look at the modern CSS toolchain I fear we are very good at making simple things complicated.

RETWEETS
271

FAVORITES
294



5:28 AM - 30 Jan 2015

Knowing your core skills
brings opportunity to
contribute

Giving back

If you have been doing this
for a year, there is someone
6 months in who you are
ideally placed to help.

You will learn by teaching

Contribute to the
standards that make up the
web

Grid by Example

A collection of usage examples for the CSS Grid Layout specification.

[What is Grid Layout?](#)[Learn Grid Layout](#)[Browser Support](#)[Resources](#)

CSS Grid Layout

The CSS Grid Layout specification is one of a few new specifications that are redefining how we approach layout for the web. Alongside Flexbox and the Box Alignment Module it will become part of a modern layout system for websites and web applications.

This site is growing collection of examples, with the aim of helping people understand how Grid Layout works.

To view any of the examples in this site you need to enable CSS Grid Layout in your browser, or

Get Started with Grid

A simple grid



The simplest example. Create a three column grid. Grid Layout will auto place items one in each cell.

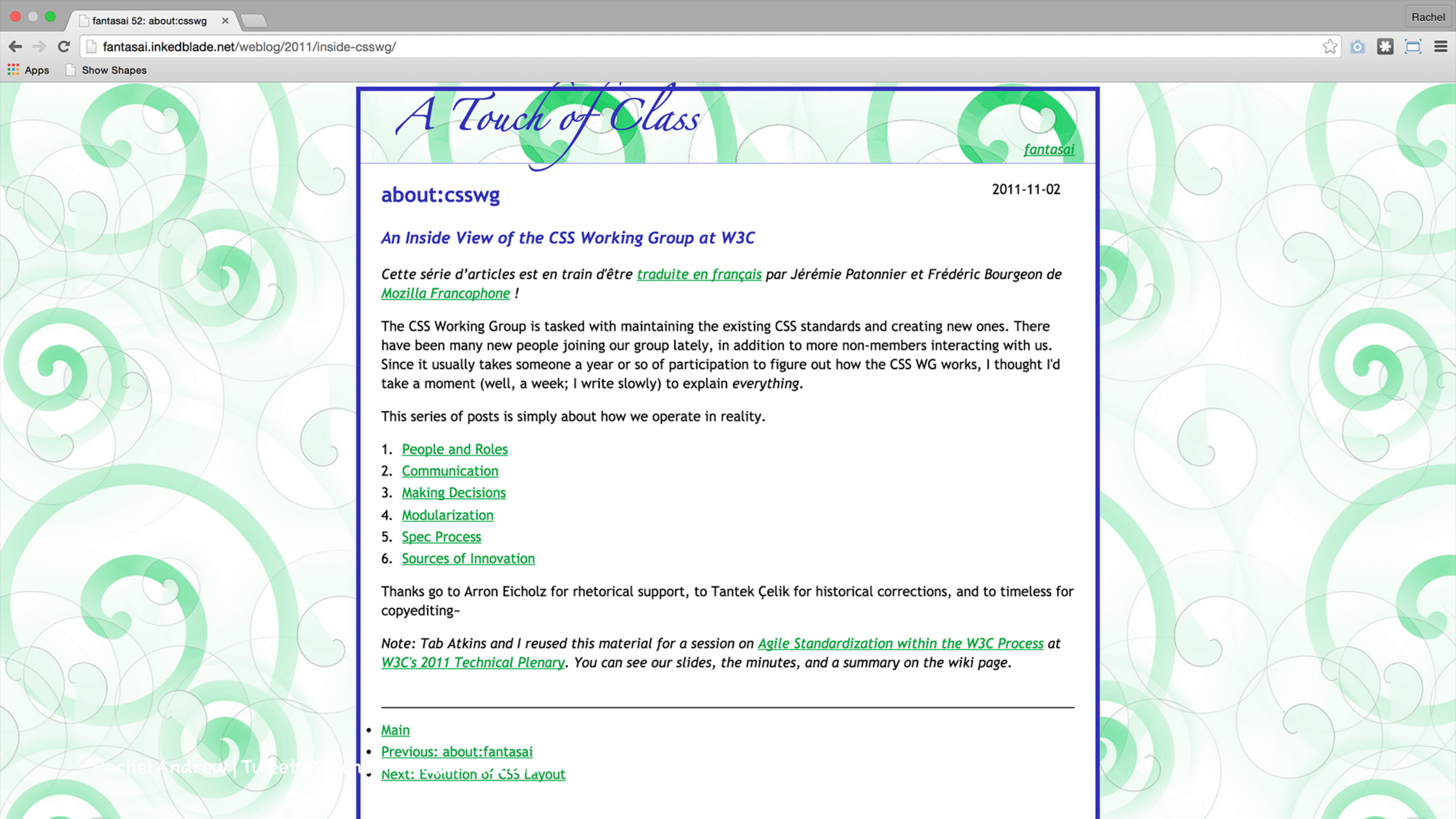
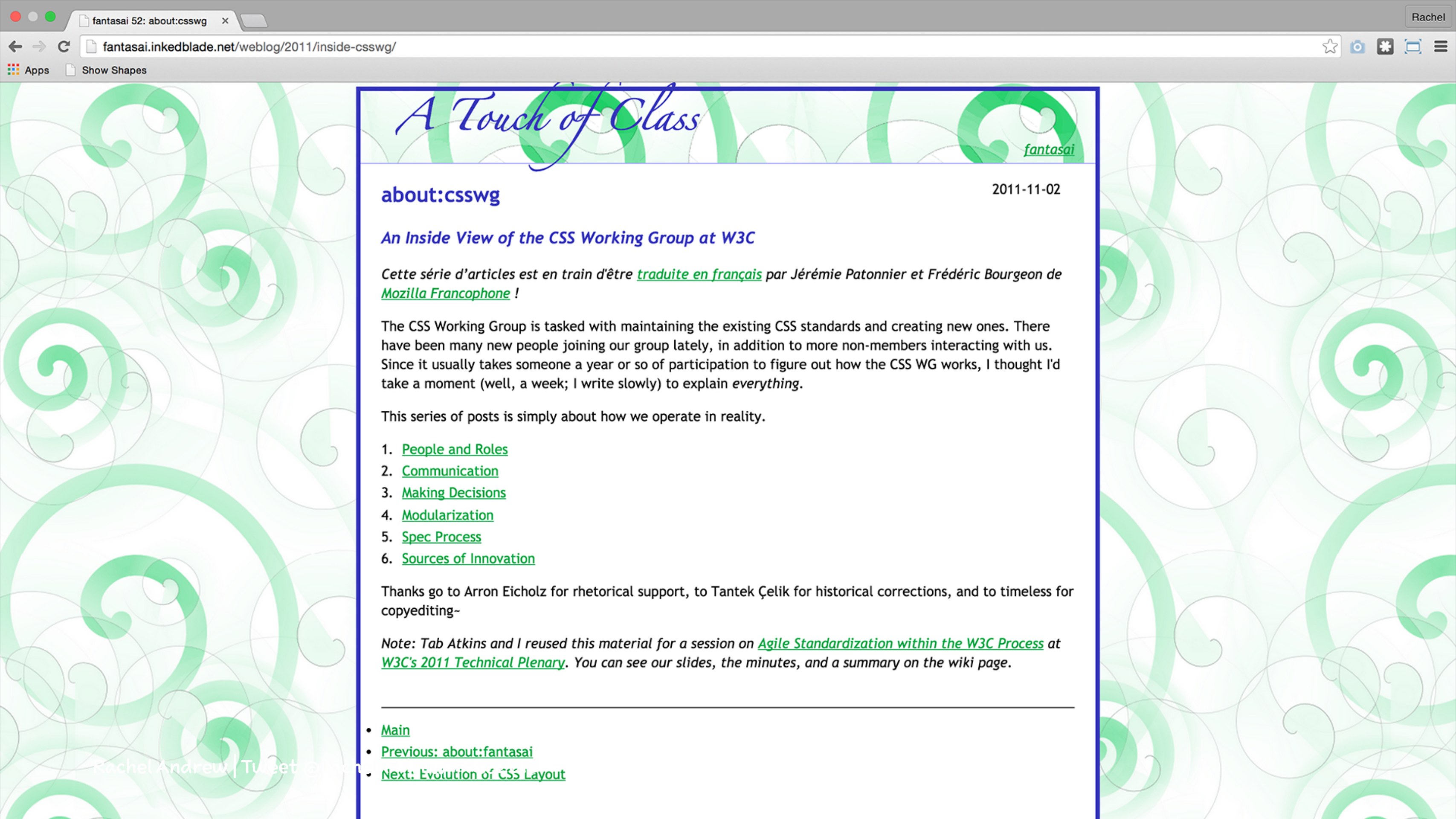
The CSS Working Group

What do authors think?

Making a difference to a CSS Specification

can be as straightforward as writing
about how you want to use it

Learn how the modern
standards process works



A Touch of Class

fantasai

about:csswg 2011-11-02

An Inside View of the CSS Working Group at W3C

Cette série d'articles est en train d'être [traduite en français](#) par Jérémie Patonnier et Frédéric Bourgeon de [Mozilla Francophone](#) !

The CSS Working Group is tasked with maintaining the existing CSS standards and creating new ones. There have been many new people joining our group lately, in addition to more non-members interacting with us. Since it usually takes someone a year or so of participation to figure out how the CSS WG works, I thought I'd take a moment (well, a week; I write slowly) to explain *everything*.

This series of posts is simply about how we operate in reality.

1. [People and Roles](#)
2. [Communication](#)
3. [Making Decisions](#)
4. [Modularization](#)
5. [Spec Process](#)
6. [Sources of Innovation](#)

Thanks go to Arron Eicholz for rhetorical support, to Tantek Çelik for historical corrections, and to timeless for copyediting~

Note: Tab Atkins and I reused this material for a session on [Agile Standardization within the W3C Process](#) at [W3C's 2011 Technical Plenary](#). You can see our slides, the minutes, and a summary on the wiki page.

-
- [Main](#)
 - [Previous: about:fantasai](#)
 - [Next: Evolution of CSS Layout](#)

To make an impact on a
specification you need to do
so while it is still
experimental

Vendor prefixes are going
away

The problem with feature flags

Developing behind flags
means authors need to be
proactive in testing

It's a **feature** not a bug

Incomplete List of Mistakes in the Design of CSS

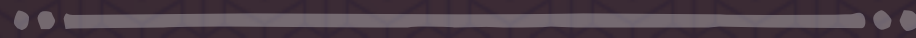
That should be corrected if anyone invents a time machine. :P

- `white-space: nowrap` should be `white-space: no-wrap`
 - and line wrapping behavior should not have been added to `white-space`
- `vertical-align` should not apply to table cells. Instead the CSS3 alignment properties should exist in Level 1.
- `vertical-align: middle` should be `text-middle` because it's not really in the middle.
- Percentage heights should be calculated against `fill-available` rather than being undefined in auto situations.
- Table layout should be sane.
- Box-sizing should be `border-box` by default.
- `background-size` with one value should duplicate its value, not default the second one to `auto`.
- `background-position` and `border-spacing` (all 2-axis properties) should take **vertical** first, to match with the 4-direction properties like `margin`.
- The 4-value shorthands like `margin` should go counter-clockwise (so that the `inline-start` value is before the `block-start` value).
- `z-index` should be called `z-order` or `depth` and should Just Work on all elements (like it does on flex items).
- `word-wrap/overflow-wrap` should not exist. Instead, `overflow-wrap` should be a keyword on `'white-space'`, like `nowrap` (`no-wrap`).
- The top and bottom margins of a box should never have been allowed to collapse together automatically as this is the **root of all margin-collapsing evil**.
- Partial collapsing of margins instead of weird rules to handle `min/max-heights`?
- Tables (and other non-blocks, e.g. flex containers) should form pseudo-stacking contexts.
- The `currentcolor` keyword should have a dash, `current-color`.
- There should have been a predictable color naming system instead of arbitrary X11 names.
- `border-radius` should have been `corner-radius`.
- Absolutely-positioned replaced elements should stretch when opposite offset properties (e.g. `left+right`) are set, instead of being start-aligned.
- The `hyphens` property should be called `hyphenate`. (It's called `hyphens` because the XSL:FO people objected to `hyphenate`.)
- `rgba()` and `hsla()` should not exist, `rgb()` and `hsl()` should have gotten an optional fourth parameter instead (and the alpha value should have used the same format as R, G, and B or S and L).
- descendant combinator should have been `>` and indirect sibling combinator should have been `++`, so there's some logical relationships among the selectors' ascii art
- the `*-blend-mode` properties should've just been `*-blend`
- The syntax of unicode ranges should have consistent with the rest of CSS, like `u0001-u00c8`.
- `font-family` should have required the font name to be quoted (like all other values that come from "outside" CSS). The rules for handling unquoted font names make parsing `font` stupid, as it requires a `font-size` value for disambiguation.
- Flexbox should have been less crazy about `flex-basis` vs `width/height`. Perhaps: if `width/height` is `auto`, use `flex-basis`; otherwise, stick with `width/height` as an inflexible size. (This also makes `min/max width/height` behavior fall out of the generic definition.)

If authors do not offer feedback

the final specification will reflect our
needs as understood by people who do
not build websites.

Contributing to the open web platform



is like giving future you a gift

Contribute to CSS Specifications

- Specifications are discussed on GitHub at <https://github.com/w3c/csswg-drafts>



This repository

Search

Pull requests

Issues

Gist



w3c / csswg-drafts

Unwatch 124

★ Unstar 258

🍴 Fork 78

<> Code

🔔 Issues 288

🔗 Pull requests 12

📁 Projects 2

📈 Pulse

📊 Graphs

[css-display] create a display property value for visually hiding an element while making it available for AT #560

New issue

🔔 Open

SaraSoueidan opened this issue 12 days ago · 6 comments



SaraSoueidan commented 12 days ago



While giving a talk at CSSConf last week, I mentioned how we should provide text for AT to be able to read when we are using only icons to represent that text visually. Basically: provide text in the DOM that screen readers can read, and then hide it visually by using one of several visually-hidden techniques/hacks that we currently use for this purpose.

After the talk, [an attendee asked](#) why we don't have a CSS property whose sole purpose would be to hide content visually while keeping it readable by screen readers, for example.

We know `display: none` and `visibility: hidden` both hide content visually but they also make it inaccessible by AT. Any chance we could get a `display` value that would hide text similar to the way `display: none` does but that also keeps the text accessible underneath?

Thanks!



AmeliaBR commented 12 days ago • edited



There isn't a CSS proposal, but there is an ARIA property, at least in theory. `aria-hidden: false` is supposed to expose content to assistive technology even if it would normally be considered hidden because of `display: none` or `visibility: hidden` or width/height of 0.

However, because browsers did not support `aria-hidden: false` consistently, accessibility best practice has developed to use a combination of properties (overflow/clip plus absolute positioning, and/or positioning off-screen) that none of the browsers use in their heuristics for deciding whether

Projects

None yet

Labels

css-display-3

Milestone

No milestone

Assignees

No one assigned

6 participants



Notifications

🔔 Unsubscribe

You're receiving notifications because you're subscribed to this repository.

Contribute to interoperability

Raise bugs with browsers

Learn to create a reduced test case



This is a skill that will save time in
your own work and also in logging
issues with any project

Keep learning

Learning. Contributing

Excited about the future.

Thank you!

Slides and links at
<https://cssgrid.me/ato-keynote>

@rachelandrew