IN THE UNITED STATES DISTRICT COURT
FOR THE EASTERN DISTRICT OF TEXAS
MARSHALL DIVISION

SoftView, LLC, Plaintiff, v. Samsung Electronics Co., Ltd. and Samsung Electronics America, Inc., Defendants.	Case No.: 2:25-cv-00246 **JURY TRIAL DEMANDED**

COMPLAINT FOR PATENT INFRINGEMENT

Plaintiff, SoftView, LLC, (hereafter "SoftView" or "Plaintiff") files this complaint under 35 U.S.C. § 271 against Defendants Samsung Electronics Co., Ltd. ("SEC") and Samsung Electronics America, Inc. ("SEA") (collectively "Defendants" or "Samsung") for infringement of U.S. Patent Nos. 9,519,729, 7,844,889, 10,083,154, 8,533,628, and 7,461,353 (the "Asserted Patents"). SoftView, on personal knowledge as to its own acts, and upon information and belief as to all others based on investigation, alleges as follows:

NATURE OF THE ACTION

1. This is a civil action for patent infringement arising under the United States Patent Act, 35 U.S.C. § 101 *et seq*., including 35 U.S.C. § 271. This Court has subject matter jurisdiction pursuant to 28 U.S.C. §§ 1331 and 1338(a) in that this is a civil action arising out of the patent laws of the United States of America.

PARTIES

2. Plaintiff SoftView is a limited liability company organized and existing under the laws of the State of Washington, with a principal place of business at 3112 Maple Ridge Court, Bellingham, Washington 98229.

GOOGLE EXHIBIT 1058
Google LLC v. Softoview LLC
IPR2026-00281

3. SoftView is the true and correct owner of the Asserted Patents and holds all rights necessary to bring this action.

4. On information and belief, Defendant Samsung Electronics Co. Ltd. ("SEC") is a corporation organized under the laws of the Republic of Korea, with a principal place of business at 129, Samsung-ro, Yeongtong-gu, Suwon-si, Gyeonggi-do, South Korea.

5. On information and belief, Defendant Samsung Electronics America, Inc. ("SEA") is a wholly owned subsidiary of Samsung Electronics Co., Ltd. and a limited liability company organized under the laws of New York, with a principal place of business at 85 Challenger Road, Ridgefield Park, New Jersey 07660.

6. On information and belief, Defendant SEC owns one hundred percent (100%) of SEA.

7. Joinder is proper under 35 U.S.C. § 299. The allegations of infringement contained herein are asserted against the Defendants SEC and SEA (collectively, "Defendants") jointly, severally, or in the alternative and arise, at least in part, out of the same series of transactions or occurrences relating to Defendants' manufacture, use, sale, offer for sale, and importation of the same accused products.

8. On information and belief, Defendants are part of the same corporate family of companies, and the infringement allegations arise at least in part from Defendants' collective activities with respect to Defendants' accused products. Questions of fact common to Defendants will arise in the action, including questions relating to the structure and operation of the accused products and Defendants' infringing acts.

THE PATENTS IN SUIT

9. On December 13, 2016, United States Patent No. 9,519,729, entitled "Scalable Display of Internet Content on Mobile Devices" ("the '729 patent"), was issued to Gary B.

Rohrabaugh and Scott A. Sherman. SoftView is the owner by assignment of the entire right, title, and interest in and to the ’729 patent. A copy of the ’729 patent is attached as **Exhibit A**.

10. On November 30, 2010, United States Patent No. 7,844,889, entitled “Resolution Independent Display of Internet Content” (“the ’889 patent”), was issued to Gary B. Rohrabaugh and Scott A. Sherman. SoftView is the owner by assignment of the entire right, title, and interest in and to the ’889 patent. A copy of the ’889 patent is attached as **Exhibit B**.

11. On September 25, 2018, United States Patent No. 10,083,154, entitled “Scalable Display of Internet Content on Mobile Devices” (“the ’154 patent”), was issued to Gary B. Rohrabaugh and Scott A. Sherman. SoftView is the owner by assignment of the entire right, title, and interest in and to the ’154 patent. A copy of the ’154 patent is attached as **Exhibit C**.

12. On September 10, 2013, United States Patent No. 8,533,628, entitled “Method, Apparatus, and Browser to Support Full-Page Web Browsing on Hand-Held Wireless Devices” (“the ’628 patent”), was issued to Gary B. Rohrabaugh and Scott A. Sherman. SoftView is the owner by assignment of the entire right, title, and interest in and to the ’628 patent. A copy of the ’628 patent is attached as **Exhibit D**.

13. On December 2, 2008, United States Patent No. 7,461,353, entitled “Scalable Display of Internet Content on Mobile Devices” (“the ’353 patent”), was issued to Gary B. Rohrabaugh and Scott A. Sherman. SoftView is the owner by assignment of the entire right, title, and interest in and to the ’353 patent. A copy of the ’353 patent is attached as **Exhibit E**, along with a Certificate of Correction, an Inter Partes Review Certificate, and an Ex Parte Reexamination Certificate for the ’353 patent.

JURISDICTION AND VENUE

14. This is an action for patent infringement under the Patent Laws of the United States, 35 U.S.C. § 1 *et seq*., over which this Court has subject matter jurisdiction pursuant to 28 U.S.C. §§ 1331 and 1338(a).

15. This Court has both general and specific jurisdiction over Defendants because Defendants have committed acts within this District giving rise to this action and have established minimum contacts with this forum such that the exercise of jurisdiction over Defendants would not offend traditional notions of fair play and substantial justice. Defendants, directly and through subsidiaries and intermediaries (including distributors, retailers, franchisees, and others), have committed acts of patent infringement in this District, by, among other things, making, using, testing, selling, licensing, importing and/or offering for sale/license products and services that infringe the SoftView Patents.

16. The Court has personal jurisdiction over Defendants for at least the following reasons: (1) Defendants have committed acts of patent infringement in this District and elsewhere in Texas; (2) Defendants regularly do business or solicit business in this District and in Texas, including through their wholly-owned subsidiary entities; (3) Defendants engage in other persistent courses of conduct and derive substantial revenue from products and/or services provided to individuals in this District and in Texas; (4) Defendants have purposefully established substantial, systematic, and continuous contacts with this District, including the physical location at 6625 Excellence Way, Plano, TX 75023, and should reasonably expect to be subject to suit in this District; and (5) Defendant SEA has been authorized to do business in the State of Texas by the Texas Secretary of State.

17. Defendant SEC is a foreign entity such that venue is proper in any District within the United States.

18. Venue is proper in this District under 28 U.S.C. §§ 1391(b)-(d) and 1400(b). Defendant SEA is subject to personal jurisdiction in this District, regularly conducts business in this District having an established place of business in this District at 6625 Excellence Way, Plano, TX 75023, and has committed acts of direct and indirect patent infringement in this District.

19. Defendant SEA has designated CT Corporation System, with an address listed in Texas at 1999 Bryan St., Ste. 900, Dallas, TX 75201, as its representative to accept service of process within the State of Texas.

20. On information and belief, Defendants have at least hundreds of employees within this District. On information and belief, third-party witnesses with knowledge regarding the accused functionality reside within this District.

21. Defendants have committed tortious acts within Texas and this District, and the causes of action set forth in this Complaint arise from those acts. Defendants developed, manufactured, imported, distributed, and sold mobile telephone and computing products that infringed the Asserted Patents, which have been imported, offered for sale, sold (directly or through Defendants' online store and distribution network), purchased, and used in Texas and within this District. Defendants also placed infringing products within the stream of commerce, with the knowledge and/or understanding that such infringing products would be sold and/or used in Texas and in this District.

PRIOR LITIGATION OF THE '353 AND '926 PATENTS

22. On May 10, 2010, SoftView filed an assertion for patent infringement of the '353 patent against Apple and AT&T in the District of Delaware: *SoftView LLC v. Apple Inc. et al.*, No. 1:10-cv-00389-LPS (Judge Stark). In December 2010, SoftView filed an amended complaint to add United States Patent No. 7,831,926 (the '926 patent), which issued on November 9, 2010. In

April 2011, SoftView filed a motion to add several Android device manufacturers, including HTC, Huawei, LG, Samsung, Dell, Kyocera, Motorola Mobility, and Sony Ericsson. The motion was granted on September 30, 2011. Subsequently, on July 26, 2012, the litigation was split into the following 9 cases:

1:10-cv-00389-LPS	SoftView LLC v. Apple Inc. et al	filed 05/10/10	closed 10/29/15
1:12-cv-00984-LPS	SoftView LLC V. HTC Corp. et al	filed 07/26/12	closed 10/29/15
1:12-cv-00985-LPS	SoftView LLC v. Huawei Technologies Co. Ltd. et al	filed 07/26/12	closed 10/29/15
1:12-cv-00986-LPS	SoftView LLC v. LG Electronics Inc. et al	filed 07/26/12	closed 10/29/15
1:12-cv-00987-LPS	SoftView LLC v. Samsung Electronics Co. Ltd. et al	filed 07/26/12	closed 10/29/15
1:12-cv-00988-LPS	SoftView LLC v. Dell Inc. et al	filed 07/26/12	closed 10/29/15
1:12-cv-00989-LPS	SoftView LLC v. Kyocera Corp. et al	filed 07/26/12	closed 10/29/15
1:12-cv-00990-LPS	SoftView LLC v. Motorola Mobility LLC et al	filed 07/26/12	closed 10/29/15
1:12-cv-00991-LPS	SoftView LLC v. Sony Ericsson Mobile Communications (USA) Inc. et al	filed 07/26/12	closed 10/29/15

23. During the litigation, Kyocera filed an *Inter Partes* Review (IPR) against each of the ’353 and ’926 patents, which were later joined by Motorola Mobility LLC. In the ’353 Final Written Decision (FWD), claims 1, 33, 36, 43, 48, 51, 52, 58, 59, 66, 118, 138, 139, 149, 182, 252,

283, and 317 of the ’353 patent were found invalid and cancelled in IPR2013-00007 & IPR2013-00256, affirmed by *Softview LLC v. Kyocera Corp. and Motorola Mobility LLC.*, No. 2014-1600 (Fed. Cir. Feb. 9, 2015, Rule 36 Judgment). In the ’926 FWD, claims 30, 31, 40, 41, 43, 52, 55, 59, 72 and 75 were found invalid and cancelled in IPR2013-00004 & IPR2013-00257, affirmed by *Softview LLC v. Kyocera Corp. and Motorola Mobility LLC.*, No. 2014-1599 (Fed. Cir. Feb. 9, 2015, Rule 36 Judgment). The claims cancelled in the IPRs included the 20 claims asserted by SoftView.

24. The litigation was dismissed on October 29, 2015 as applied to all Defendants, including Samsung Electronics Co. Ltd. et al. In an oral order issued by Judge Stark, the litigation for 1:12-cv-00987-LPS was DISMISSED WITH PREJUDICE with respect to the 20 asserted claims SoftView identified in accord with the Court’s case management orders, and DISMISSED WITHOUT PREJUDICE as to all other claims (*i.e.*, all but the 20 asserted claims SoftView identified in accord with the Court’s case management orders).

Docket Text:

ORAL ORDER: Having reviewed the parties joint status report (D.I. 1125), as well as the previously-submitted status report and attachments thereto (D.I. 1117), IT IS HEREBY ORDERED that: (1) the STAY is LIFTED; (2) this matter is DISMISSED WITH PREJUDICE with respect to the 20 asserted claims SoftView identified in accord with the Courts case management orders; and (3) this matter is DISMISSED WITHOUT PREJUDICE as to all other claims (*i.e.*, all but the 20 asserted claims SoftView identified in accord with the Courts case management orders). The Clerk of Court is directed to CLOSE this case. ORDERED by Judge Leonard P. Stark on 10/29/15. Associated Cases: 1:10-cv-00389-LPS et al. (ntl)

Defendants had requested that the entire case be dismissed with prejudice.

BACKGROUND

History of Mobile Browsers

25. Tim Berners-Lee, the inventor of the first "WorldWideWeb" browser, originally envisioned Web browsers enabling users around the world access to text-only documents independent of the user's computer systems. Berner-Lee's browser was a universal line-mode browser that was designed to work simply by typing commands. There was no mouse or graphics; rather, the core principle of the web was that content should be universally accessible. There was also no concept of page layout or design, as text content was designed to wrap within the width of the computer display or browser window. This simplistic approach was employed by early mobile browsers, such as the NetHopper browser on the Apple Newton and the first browsers on PalmPilots. These were text-only browsers, with very limited formatting (*e.g.*, support for headers and paragraphs, but no columns, tables, frames, etc.).

26. Beginning with the public release of the Mosaic Web browser (the first significant graphical Web browser developed at the National Center for Supercomputing Applications (NCSA)), desktop Web browsers and W3C Web standards advanced dramatically from 1993-2001, including HTML 2, 3.2, and 4/4.01, Cascading Style Sheets 1.0 (CSS1) and CSS2, XML, and DOM (Document Object Model). Meanwhile, mobile browsers, which had a later start, were left behind. While partially due to limited processing and memory resources available on the mobile devices, the principal reason was the small screen size and limited resolution. The approach used by many early mobile browsers was to "adapt" presentation of the content to the display capabilities of the device. Since the device's resources were so limited, adaptation operations were typically performed via a proxy server or gateway. Under one approach, desktop Web pages were adapted by stripping out and/or simplifying content, such as images, and

reformatting the layout of the content to fit the device's display parameters. Another approach was Wireless Application Protocol (WAP). Under this approach, HTML-based Web pages were processed by a WAP gateway to generate WML (Wireless Markup Language) content, which completely obliterated the original page layout and design, and typically stripped out a significant amount of content, including images. Optionally, original WAP pages could be created from scratch. In addition to WAP, there were other non-HTML approaches, such as cHTML and iMode in Japan and Openwave's HDML (Handheld Device Markup Language).

27. During the late 1990's, HTML-based browsers employing stripped down rendering engines became available, such as for some devices in Japan and devices running Microsoft Windows CE-based operating systems. These browsers provided limited or no support for HTML 3.2, 4/4.01, CSS, and/or XML, resulting in pages being rendered incorrectly and/or missing content or functionality. While these crippled HTML-based browsers had difficulties with desktop pages, they lead to the development of mobile-only Web pages that were written in HTML and designed for a target display width resolution (or using a free-flow layout) to present a single column of content across the width of the device's display.

28. None of the foregoing approaches were satisfactory. Mobile browsing was generally a painful experience on several fronts—it was very slow, the adaptation approaches produced pages with missing content that looked nothing like desktop pages, and WAP pages were even worse. Viewing full desktop pages designed for screen resolutions much greater than the device's resolution was very cumbersome, and while mobile-only pages were an improvement on some levels, there were few Web sites that provided such pages, and the content of these pages was rather limited.

Brief History of SoftSource

29. Gary Rohrabaugh and Scott Sherman worked together at SoftSource, founded in 1982 in Bellingham, WA, and invented SoftView's patented technology. From the mid-1980's through the 1990's, SoftSource developed software relating to Computer Aided Design (CAD) and computer graphics. In the earlier years this included award-winning software such as Drawing Librarian, Block Librarian and Drawing Exchange Engine (DXE), which were designed to view, translate, and edit AutoCAD drawings, the world's most-used CAD software. After a joint project with Autodesk to rewrite AutoCAD collapsed in 1992, SoftSource began development of its own CAD system, VDraft (Virtual Drafter), which was released in 1997. In 1993, SoftSource began development of Simple Vector Format in conjunction with NCSA, leading to SVF 1.0. SVF was proposed to the W3C as a standard for delivering vector-based content across the Internet. SoftSource developed and demonstrated an Internet-based multi-user version of VDraft that employed SVF in 1999. In 1999, SoftSource also developed an OEM product to support viewing maps on mobile devices for ZoomOn (of Swedish phone company Telesis).

30. Mr. Rohrabaugh had been working with HTML since 1993 and observed Web pages advance from simple free-flow layout of text content to sophisticated and highly-structured page layouts that were designed to be displayed at a fixed resolution, such as 640 x 480 (VGA), and later 800 x 600 and 1024 x 768. The introduction of HTML 4/4.01 and CSS added even further design and layout capabilities. While these advancements were great, they often created headaches for Web page designers. In particular, how do you create a universal design such that a page will look good and be usable with monitors and devices having various resolutions and screen sizes? The reality was that designing a page to have good appearance and usability under one set of parameters resulted in poor appearance and/or usability under other parameters. One solution was to design pages for different resolutions, typically by reformatting layout and content. However,

the layout and design of pages such as Yahoo!, New York Times, ESPN, CNN, *etc.* became a *de facto* branding for the pages and were so familiar that a user could immediately navigate to where she wanted to go; reformatting these pages defeated both branding and usability. Supporting multiple page designs also requires substantial Web designer resources; as a result, only a minute fraction of Web sites could support this approach.

Genesis of the Patented Inventions

31. Following development of the original version of SVF in 1993, Mr. Rohrabaugh began to conceive of ways he could apply SVF and its associated concepts to the Internet at a broader level.

32. These original concepts would enable users to view Internet content on emerging portable devices, with hyperlink support.

33. From 1994-1997, SoftSource's primary development focus was the VDraft CAD system, with the Internet concepts being put on the back burner. Toward the end of 1997, Mr. Rohrabaugh outlined "SVF CAD" to run on Windows CE devices using SVF 2.0 (which was also being defined and had yet to be implemented).

34. During 1998-1999, Mr. Rohrabaugh had been further researching applications that could leverage SoftSource's vector technology on various devices, including PDAs. This included development of a vector-based map viewer for ZoomOn (Stockholm, Sweden). In connection with the ZoomOn development, Mr. Rohrabaugh conceived of a broader vision – enable users of mobile devices to browse the entire Internet by making HTML zoomable. This was such a novel and revolutionary idea that even members of the development team at SoftSource initially did not believe it was possible, while others outside of SoftSource insisted this could not be done. Indeed, in 1999 it was not possible, as the necessary software and hardware pieces had to be developed by SoftSource, Mozilla, and others and integrated into a viable solution.

35. Having observed two decades of computer graphics evolution and through personal experience designing Web pages, Mr. Rohrabaugh recognized that the problems with the existing approaches were only going to get worse. Future computers and devices would employ higher resolution screens with the size of screens would be more varied, from very small screens on handhelds to widescreen video monitors to large media boards. Moreover, the pixel resolution density would continue to increase. Rohrabaugh envisioned that by enabling the page content to be viewed in a resolution-independent manner while preserving the page's original layout, functionality, and design at different zoom levels, billions of existing Web pages could be browsed on computers and devices having substantially any screen resolution and size. Furthermore, as a user zoomed in, the effective resolution of the display content should increase, and zooming and panning operations should be affected in real-time using simple user inputs and gestures.

Development of the Solution

36. In January 2000, Mr. Rohrabaugh formed ZHTML.com, where 'Z' stood for Zoomable. The main goal of this company was to define, build, and deploy vector-based Internet page view technology called ZHTML. ZHTML was intended to be the next generation of vector technology that built SoftSource's prior successes, including managing and implementing the first vector display system for the Web, SVF. SoftSource was one of a handful of companies granted a free source code license to Mosaic (in 1994). In ZHTML, SoftSource would be capitalizing on its extensive experience in vector and display list software and Web development to bring a major innovation to the Internet. Mr. Rohrabaugh envisioned vector-based Web page display would eliminate one of the last great barriers to viewing and working with the Internet anywhere and on any device.

37. The SoftSource developers and software engineers investigated developing their own Web browser rendering engine and realized that developing software to support emerging

Web standards such as HTML 4/4.01, CSS 1 and CSS2, and supporting other functionality such as JavaScript and XML would be exceedingly complex, something that would take their small team many years to successfully implement, if at all. They then considered using existing rendering engines. At the time the dominant Web browsers were Microsoft Internet Explorer (IE) and Netscape Navigator. IE employed a closed-source rendering engine (called "Trident"). To have a chance of competing against IE using a browser supporting these emerging Web standards, Netscape open-sourced its browser software code on March 31, 1998. This coincided with the launch of Mozilla, an open-source software community founded by members of Netscape. Mozilla would enable Netscape to expand its development teams with little cost, leveraging volunteer unpaid developers to assist in its development efforts.

38. The developers/software engineers at Netscape initially tried to extend the functionality of their existing rendering engine, now known as the Netscape "Classic" rendering engine, but they realized this approach not viable. Netscape scrapped the Netscape "Classic" rendering engine in favor of a new rendering engine originally called the "Raptor," then renamed "NGLayout" (next generation layout) and subsequently rebranded as "Gecko," for which development began in 1997. This rendering engine is referred to today as the Mozilla rendering engine or the Mozilla Gecko rendering engine.

39. Applying an approach somewhat analogous to that used in CAD systems, inventors Rohrabaugh and Scott Sherman designed an architecture that separates the presentation (*i.e.*, viewing and interaction) of content from the underlying content model and provided a resolution-independent representation of the Web page content. Under this approach, a browser rendering engine may be employed to interpret the Web page's HTML, CSS, JavaScript, and XML content (if present) and generate a corresponding layout (called the Frame Tree) and content model

(Document Object Model (DOM)) for the Web page. This layout and content model data is then used to generate a scalable and resolution-independent representation of the Web page that enables the Web page content to be browsed and interacted with at different zoom levels and corresponding (effective) resolutions on devices with substantially any screen resolution and size while preserving the layout, functionality, and design of the page content. SoftSource's knowledge and experience from developing SVF and VDraft were instrumental in its success in implementing its ClearView™ browser (later rebranded SoftView™). For example, SVF was leveraged, redesigned, and extended to support presentation aspects in ClearView, while VDraft's display list technology is employed for facilitating rapid zooming and panning. VDraft's object model support for user interaction with objects was used to support preservation of functionality when viewing page content at different zoom levels. Inventors Rohrabaugh and Sherman also developed additional technology to enhance ClearView's usability, including tap-based context zooming (*e.g.*, zooming on columns, images, and paragraphs) and other gesture-based user interfaces.

The SoftView Patents

40. The SoftView Patents disclose processes for translating the HTML code—including "elements such as tables, column definitions, graphic images, paragraphs" and the like––into a scalable representation for display on a mobile device. '353 Patent, 15:43-18:32. For example, FIGS. 4A-4G show translation and scaling processes for a Web page (4A) that is processed to generate HTML objects (4B) that are translated into a scalable representation (4C), and FIG. 5 illustrates a translation process.

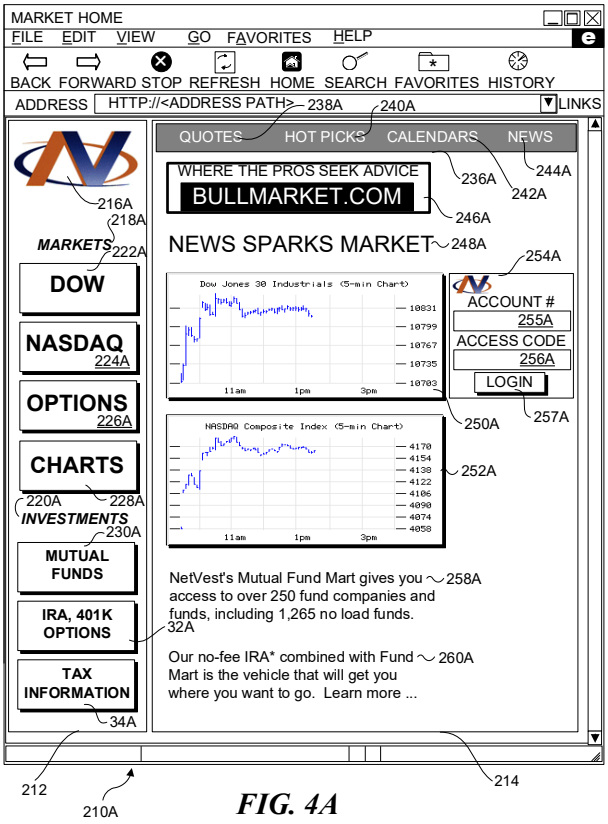


FIG. 4A

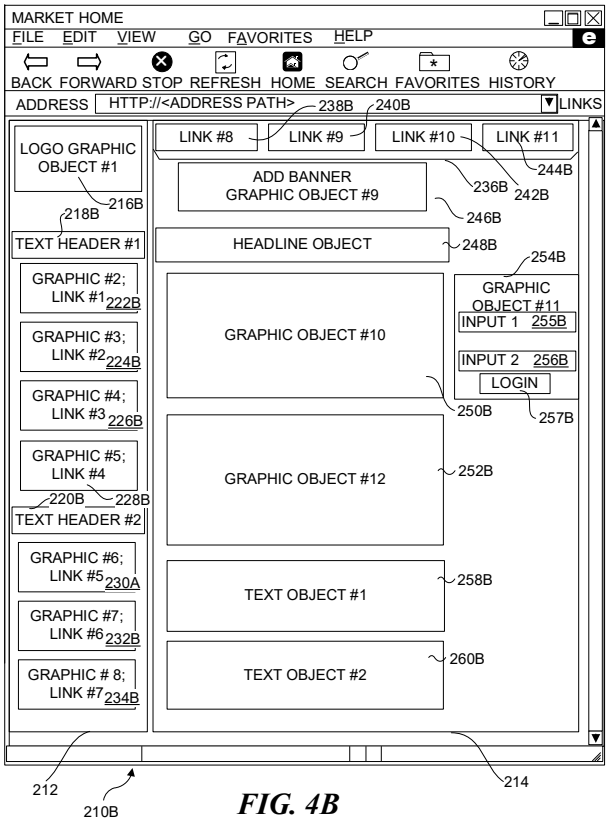


FIG. 4B

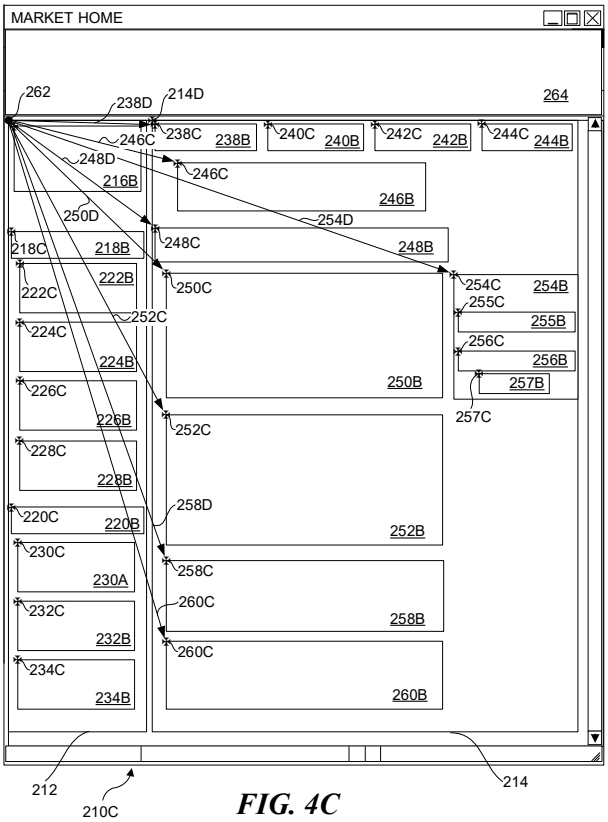


FIG. 4C

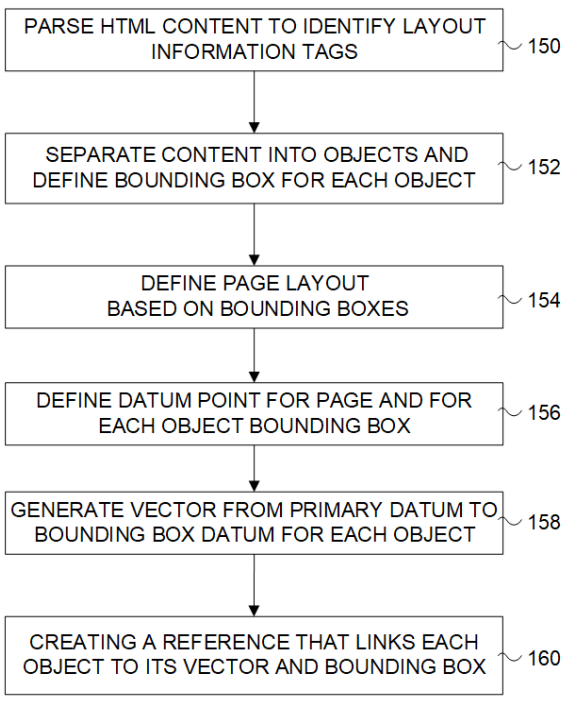


FIG. 5

41. FIGs. 4D and 4G show scaling of two HTML objects, FIG. 4E shows a scaled page, and FIG. 4F shows scaled bounding boxes for the scaled page.

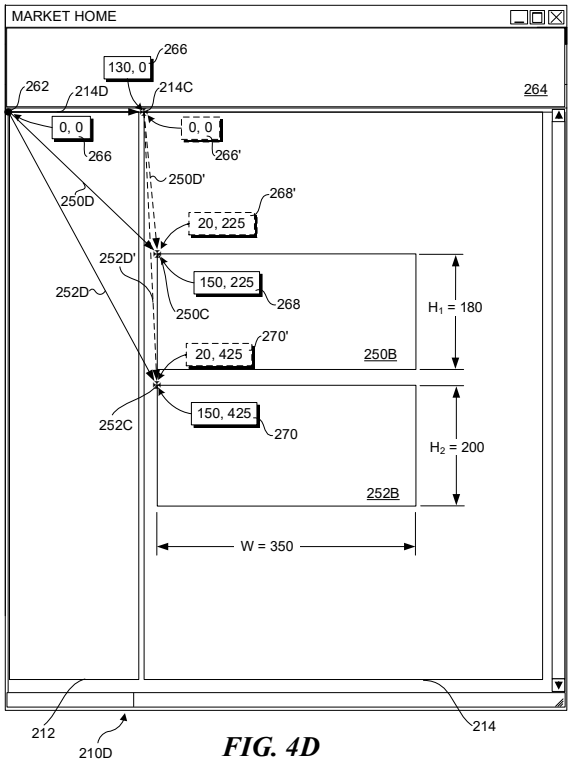


FIG. 4D

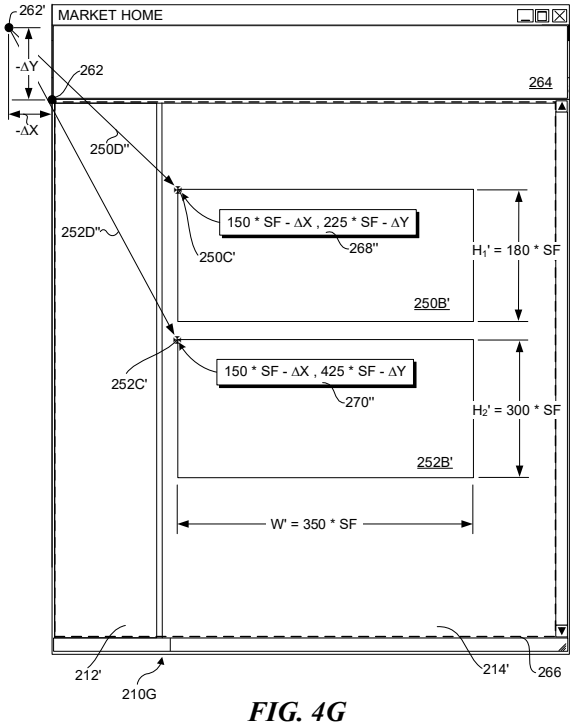


FIG. 4G

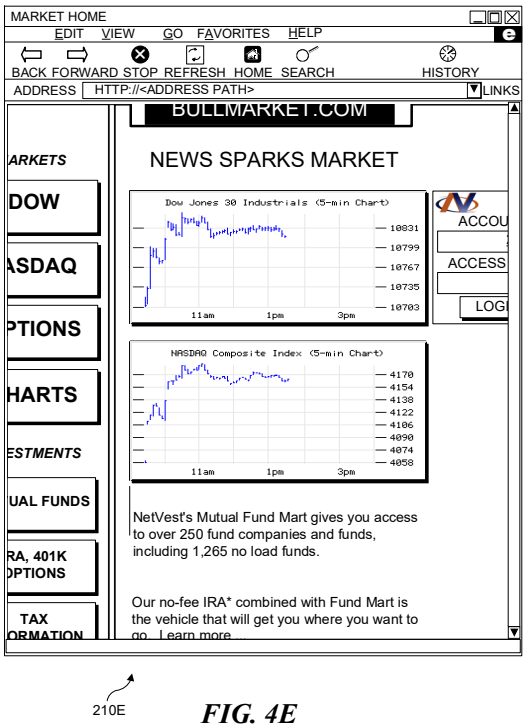


FIG. 4E

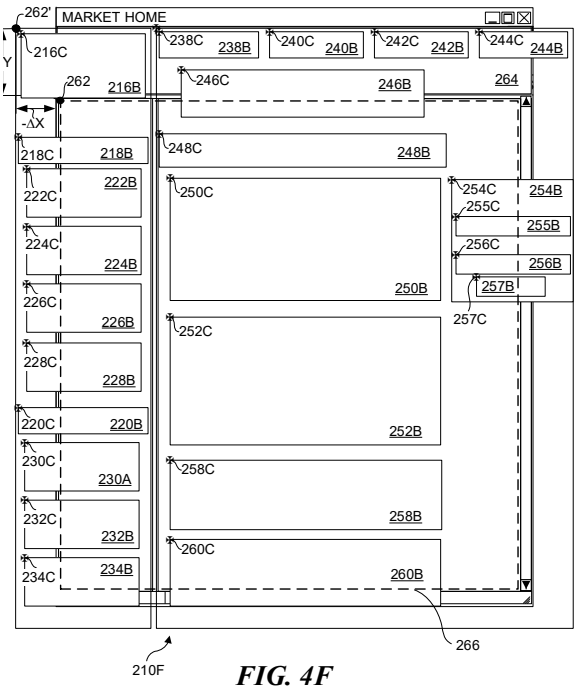


FIG. 4F

42. In exemplary embodiments, a Web page's HTML-based content is processed using an HTML- and CSS-compliant rendering engine to perform the functions in blocks 150, 152, and 154 of FIG. 5. The various object layout data is used to generate a scalable vector representation of the original page content.

43. The result was an HTML- and CSS-compliant mobile browser solution that "maintain[s] the look and feel of browsing . . . [web] pages with a conventional desktop browser" ('353 Patent, 2:54-56), and enabled users "to view the entire content of billions of existing Web pages using hand-held devices in a simple and reasonable way" ('353 Patent, 19:1-3).

44. The SoftView mobile browser solution combined the integration of highly complex software with novel user interface functionality including supporting real-time zooming and panning of Web page content while preserving the original page layout, functionality, and design of Web pages as defined by the page's content, replicating a desktop browsing experience on mobile devices. The solution also supported tap-based smart zooming on images, columns, and paragraphs.

45. FIG. 1 illustrates an overview of an exemplary process performed by the SoftView mobile browser solution and included in the File History of the '353 Patent. To access the Web page, the browser initiates an HTTP connection with the Web server hosting the Web page and begins downloading the parent HTML document.

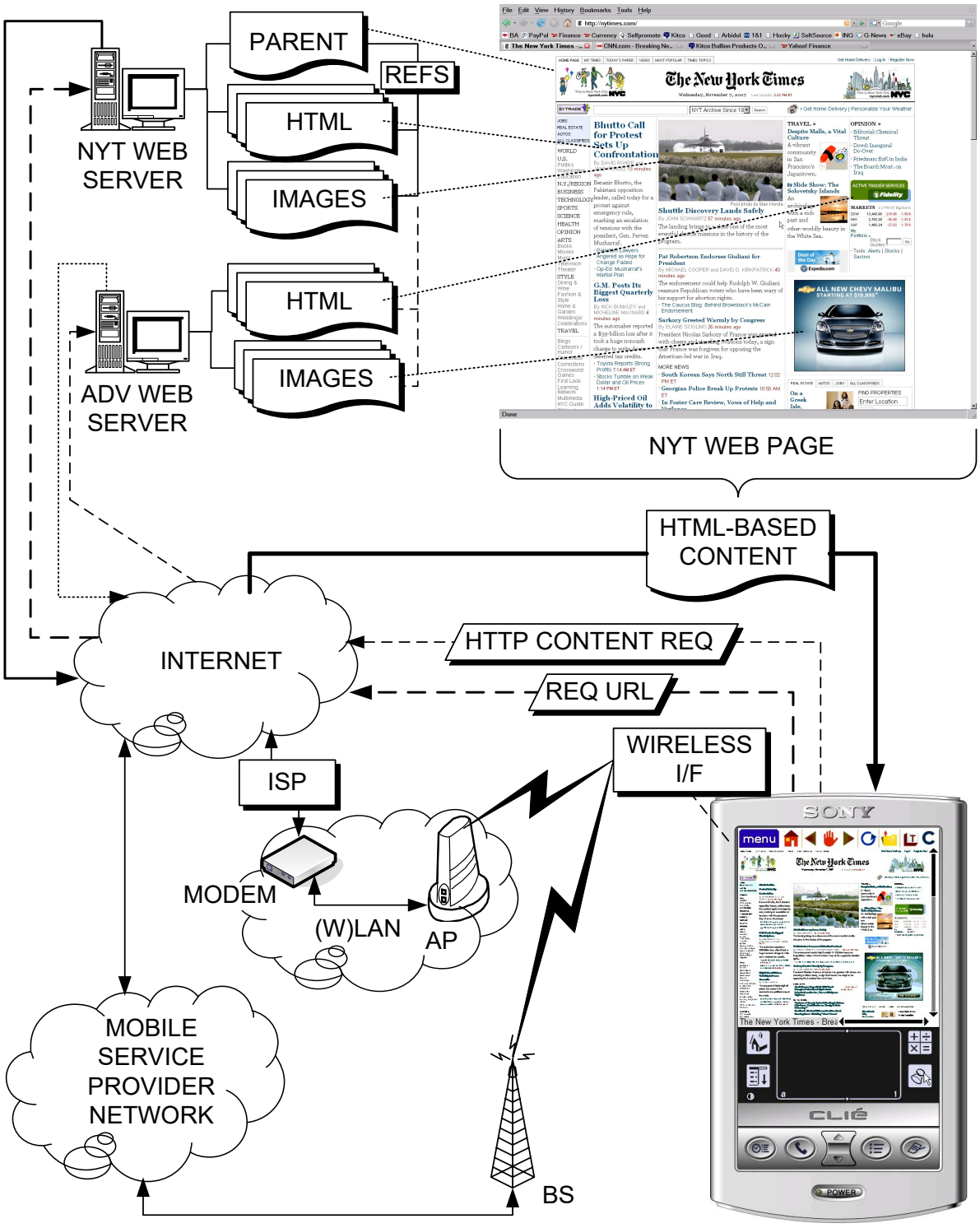


FIG. 1

46. Depending on how the Web server and/or Web page is configured, additional content referenced by the parent HTML document may be retrieved by the Web page host server

and then downloaded to the requesting client device, or a portion of this content may be downloaded by the client device via a separate connection(s).

47. The following images show examples of a Web page (www.nytimes.com November 12, 2007) rendered using a Firefox browser and how the page is rendered on a Sony Clié running Palm Operating System (OS) (1A) and a Pocket PC running Windows CE OS (1B).



Sony Clié running Palm OS	Pocket PC running Windows CE OS
	
Image 1A	**Image 1B**

SAMSUNG'S INFRINGING PRODUCTS AND ACTIVITIES

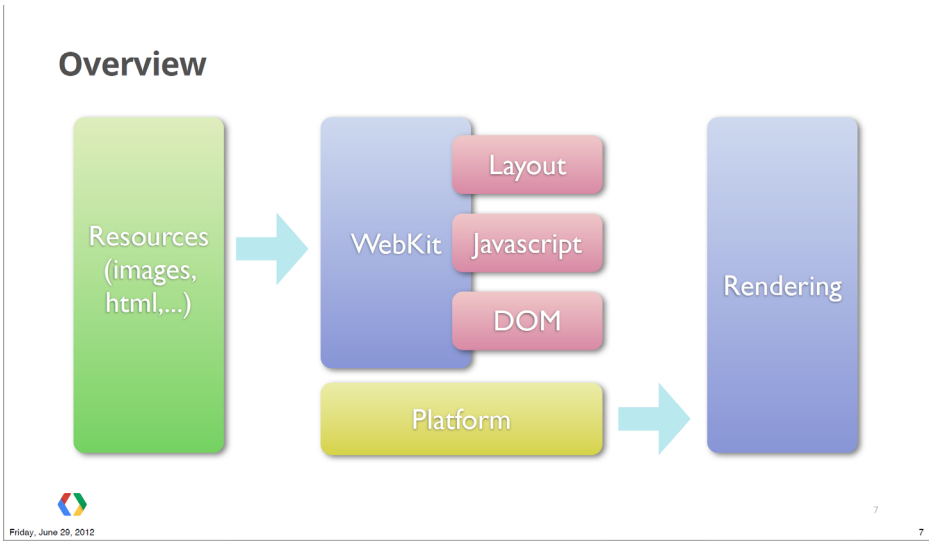
49. All Samsung smartphones and tablets since at least 2012 (collectively, "Accused Products") include at least one of the Samsung Internet browser application and the Android Chrome Web browser, which is part of the Android operating system (or was otherwise bundled with the Android operating system) since Android 4.2. Both Web browsers employ the same WebKit rendering engine code (using the "Blink" fork of WebKit/WebCore since July 10, 2013). Prior to Android 4.2, Samsung smartphones and tablets included the Samsung Internet application.

The Samsung Internet application is a modified version of the "stock" Web browser that came with the generic version of Android source code.

50. The infringing Samsung smartphones include at least the Galaxy S22, Galaxy S22 5G, Galaxy S22+ 5G, Galaxy 22 Ultra 5G, Galaxy S21, Galaxy S21 5G, Galaxy S21+ 5G, Galaxy 21 Ultra 5G, Galaxy S20, Galaxy S20+, Galaxy S20 Ultra, Galaxy S10, Galaxy S10+, Galaxy S10e, Galaxy S10 5G, Galaxy S9, Galaxy S9+, Galaxy S8, Galaxy S8+, Galaxy S7, Galaxy S7 Edge, Galaxy S7 Active, Galaxy S6, Galaxy S6 Edge, Galaxy S6 Edge+, Galaxy S6 Active, Galaxy Note20, Galaxy Note20 Ultra, Galaxy Note10+, Galaxy Note105G, Galaxy Note9, Galaxy Note8, Galaxy Note5, Galaxy Z Flip, Galaxy Z Flip 5G, Galaxy Z Flip2, Galaxy Z Flip3, Galaxy Z Flip4, Galaxy Fold, Galaxy Z Fold2 5G, Galaxy Z Fold3, Galaxy Z Fold4, Galaxy A50, Galaxy A51, Galaxy A51 5G, and Galaxy A71 5G. The infringing Samsung tablets include at least Samsung Galaxy Tab S, Tab A, and Tab E series devices.

51. The Android operating system includes libraries and frameworks associated with the "WebKit" rendering engine. In 2013, Google began working on a fork of WebKit called "Blink," which is the version of WebKit used in Android and by both the Android Internet browser and the Android Chrome browser. Both WebKit and Blink are HTML- and CSS-compliant rendering engines (meaning they comply with applicable HTML and CSS standards, and thus render Web pages having an (original) page layout, functionality, and design as defined by the page's HTML-based Web content. Although Blink is a separate fork, much of the core code (which is part of WebCore) is the same for both WebKit and Blink.

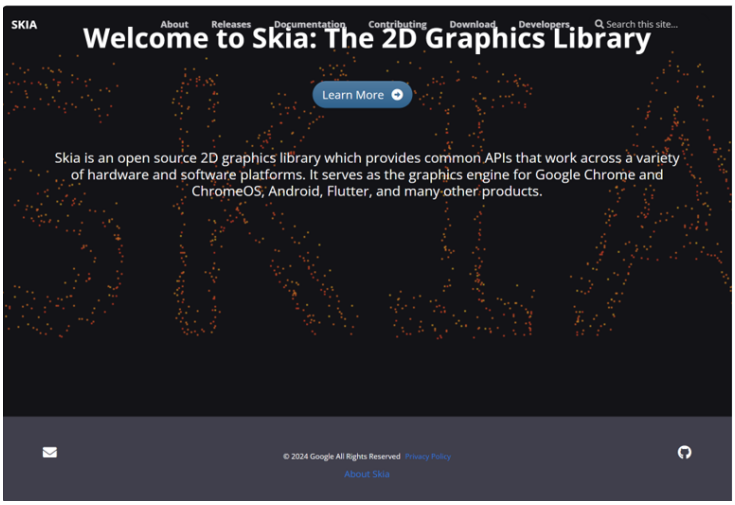
52. Android (and the Samsung Internet and Chrome browsers) use a class called "WebView" to render HTML documents, including Web pages and HTML email. A high-level architecture of WebView presented at Google IO 2012 is shown below:



https://developers.google.com/events/io/2012/sessions.

53. Android WebView (and the Samsung Internet and Chrome browsers) employs a combination of software rendering and hardware rendering. This involves use of the Skia 2D graphics library to "paint" or "rasterize" display content.

54. The Samsung Internet and Chrome browsers use the Skia 2D vector graphics rendering software/library, which is also referred to as the Skia Graphics Engine.



55. The Accused Products include software and hardware for performing the claimed methods.

COUNT I – INFRINGEMENT OF U.S. PATENT NO. 9,519,729

56. SoftView incorporates by reference and re-alleges the foregoing paragraphs of this Complaint as if fully set forth herein.

57. Defendants have directly infringed at least claim 21 of the '729 patent.

58. As a matter of law, the notice provisions of 35 U.S.C. § 287 do not apply in this case for the '729 patent, and therefore SoftView has satisfied its obligations under 35 U.S.C. § 287 with respect to the '729 patent.

59. At the time of the claimed invention, reproducing web pages on a variety of devices presented a significant technical problem. Most internet content displayed as a flat single resolution with no browser support for zoom. '353 Patent, 2:1-3.[1] Much of the content on the internet was designed for display on desktop computers with a single target resolution (e.g., 640x480, 800x600). '353 Patent, 2:4-5. Major internet content providers chose to create their Web pages using fixed resolution structures, such as tables, which gave them the ability to control the look and feel of their Web sites. '353 Patent, 2:5-10. The fixed resolution approach was so ubiquitous that it became the most common method to brand or uniquely identify websites. '353 Patent, 2:10-13.

60. While the fixed resolution approach was useful for site branding and product differentiation, it presented a daunting technical problem: how to display Internet content designed for desktop computers on devices with smaller screens, lower resolution, and/or different aspect ratios, such as cell phones and handheld computers. '353 Patent, 2:13-18. For example, a website

[1] All Asserted Patents are in the same patent family. The '353 patent is the earliest-issued Asserted Patent and is cited throughout the Complaint for consistency.

designed for the fixed resolution of a larger desktop display would not fit correctly on small screens, low resolution screens, or devices with different aspect ratios.

61. The ’729 patent claimed invention provides a solution to this daunting technical problem. Claim 21 is a dependent claim that (together with claim 14) recites as follows:

> **14.** A method, comprising:
> requesting through a mobile device having a touch-sensitive display a Web page comprising HTML-based Web content including HTML code for at least multiple columns and Cascading Style Sheets (CSS) content;
> retrieving and processing the HTML-based Web content including the HTML code for the at least multiple columns and the CSS content at the mobile device to generate a scalable vector representation of the Web page;
> rendering the Web page on the touch-sensitive display utilizing the scalable vector representation to produce a first view of the Web page at a first zoom level that preserves the original page layout, functionality and design defined by the HTML-based Web content including the at least multiple columns and the CSS content relative to the original page layout, functionality and design defined by the HTML-based Web content including the at least multiple columns and the CSS content when rendered by a commercially available desktop browser; and
> re-rendering the Web page utilizing the scalable vector representation in response to associated user inputs to iteratively zoom in and out views of the Web page at multiple different zoom levels while preserving the original page layout, functionality, and design defined by the HTML-based Web content including the multiple columns and the CSS content relative to the original page layout, functionality and design defined by the HTML-based Web content including the at least multiple columns and the CSS content when rendered by the commercially available desktop browser; and
> in response to a tap on a first of the multiple columns via the touch-sensitive display, re-rendering the Web page on the touch-sensitive display such that the first of the multiple columns is displayed to fit a width of a display area of the touch-sensitive display, the display area used for displaying the HTML-based Web content.
>
> **21.** The method of claim 14, wherein processing the HTML-based Web content comprises:
> parsing HTML-based code corresponding to the Web page to determine an original page layout of the HTML-based Web content on the Web page;
> logically grouping selected content into objects;
> defining a primary datum corresponding to the original page layout; and, for each object,
> defining an object datum corresponding to a layout location datum for the object's associated display content; and
> generating a vector from the primary datum to the object datum for the object.

62. As claimed in claim 14 of the ’729 patent, HTML-based Web content and CSS content are processed to generate a scalable vector representation of the Web page. At the time of the claimed invention, this process was not well-known, routine, or conventional. Nor was it well-known, routine, or conventional to iteratively zoom in and out views of the Web page at multiple different zoom levels while preserving the original page layout, functionality, and design defined by the HTML-based Web content including the multiple columns and the CSS content. Nor was it well-known, routine, and conventional to display a column to fit the width of a display area in response to a tap gesture.

63. Claim 21 provides additional details related to the generation of the scalable vector representation, including parsing HTML-based code corresponding to the Web page to determine an original page layout of the HTML-based Web content on the Web page; logically grouping selected content into objects; defining a primary datum corresponding to the original page layout; and, for each object, defining an object datum corresponding to a layout location datum for the object’s associated display content; and generating a vector from the primary datum to the object datum for the object. At the time of the claimed invention, this process was not well-known, routine, or conventional.

64. Claim 21 recites an object datum and primary datum that permit vectors to be defined between the object and primary datum, thereby allowing a vectorized view of the Web page. An example is shown in the exemplary Figure 4C of the patent specification, where the primary datum is in the upper left corner of the screen and each arrow points to an object datum.

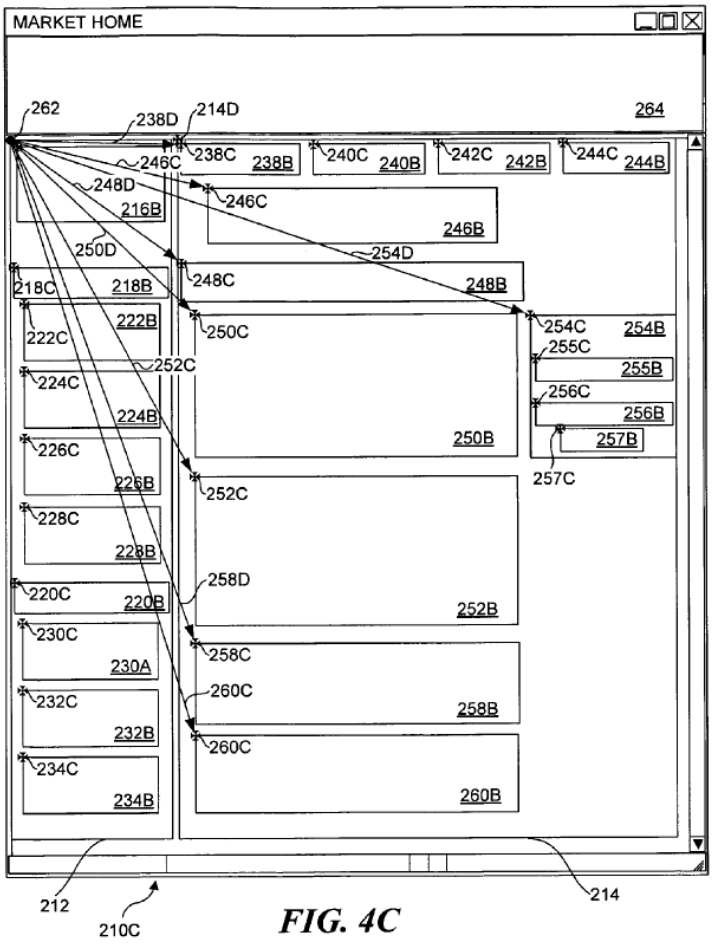


FIG. 4C

65. Claim 21 of the '729 patent is directed to a technological solution to a technological problem, and was not well-known, routine, or conventional.

66. The advantages of the '729 patent's claimed technical solution are substantial. Web pages could now "be rapidly rendered, zoomed, and panned. Moreover, the rendered displays provide substantially the same or identical layout as the original Web page, enabling users to easily navigate to selected content and features on familiar Web pages." '353 Patent, Abstract. This meant that users could finally be presented with web pages that fit and/or were effectively usable on devices with smaller screens, different resolutions, or different aspect ratios.

67. With the claimed invention, the web page could be presented as it would be on a desktop computer (as most web pages were designed for), and then rapidly panned and zoomed to

facilitate use on devices with smaller screens, different resolutions, or different aspect ratios. A Web page that was unusable when presented with the fixed resolution approach could now be zoomed to suit the device display. Today, this functionality is pervasive on smartphone and tablet devices and taken for granted. But prior to the claimed invention, it did not exist.

68. In addition, the claimed invention allowed the look and feel of the Web page to be maintained, even when designed for fixed resolutions. This allowed content providers to continue branding their Web pages with fixed resolution designs. It also allowed users to access Web pages as they were used to seeing them on desktop computers suited to their fixed resolution design, which meant users knew where to locate items on the page based on its familiar look and feel. Thus, the claimed invention made web pages more convenient for users.

69. The claimed invention has an additional advantage: internet content providers can rely on the fixed resolution approach instead of designing numerous different versions of the same web page to suit the screen size, resolution, and aspect ratio of different devices. For example, a web page designed for a 16-inch desktop display could be effectively viewed and used on a small hand-held mobile phone. This is superior to the alternative approach of designing multiple versions of the same web page to suit different devices, which was time consuming, costly, and impractical for many Web content developers.

70. The claimed invention also allowed gesture-based user inputs, such as fitting a column to the width of a display in response to a user tapping the column on the screen.

71. Defendants have directly infringed (literally and/or under the doctrine of equivalents) at least claim 21 of the '729 patent in violation of 35 U.S.C. § 271 *et seq*., through use, including testing of the Accused Products without authority or license of the Accused Products.

72. Claim 21 of the ’729 patent depends from claim 14.

73. Claim 14 recites, “A method comprising.” To the extent the preamble is limiting, the Accused Products perform a method as further described in the following paragraphs.

74. Claim 14 recites “requesting through a mobile device having a touch-sensitive display a Web page comprising HTML-based Web content including HTML code for at least multiple columns and Cascading Style Sheets (CSS) content.” The Accused Products perform this operation.

75. The Accused Products are mobile devices having a touch-sensitive display.



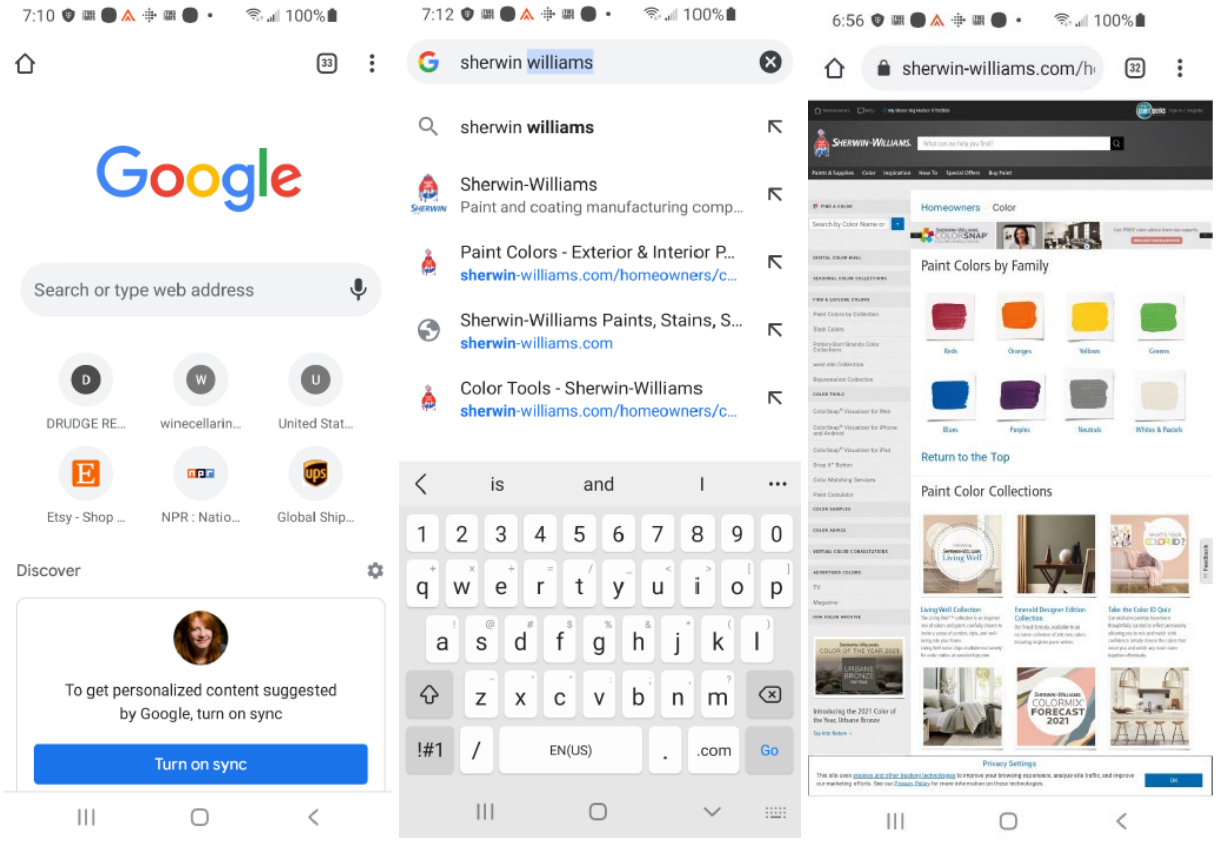
https://www.gsmarena.com/samsung_galaxy_s10-pictures-9536.php.

76. HTML (Hypertext Markup Language) and Cascading Style Sheets (CSS) employ standardized specifications for how a Web page is to appear and function. HTML is the standard markup language for documents designed to be displayed in a web browser. HTML describes the structure of a web page semantically and originally included cues for the appearance of the document. Cascading Style Sheets (CSS) is a style sheet language used for describing the presentation of a document written in a markup language such as HTML. CSS is designed to enable the separation of presentation and content, including layout, colors, and fonts. The original

page layout, functionality, and design of content of the Web page is defined by the HTML-based Web content.

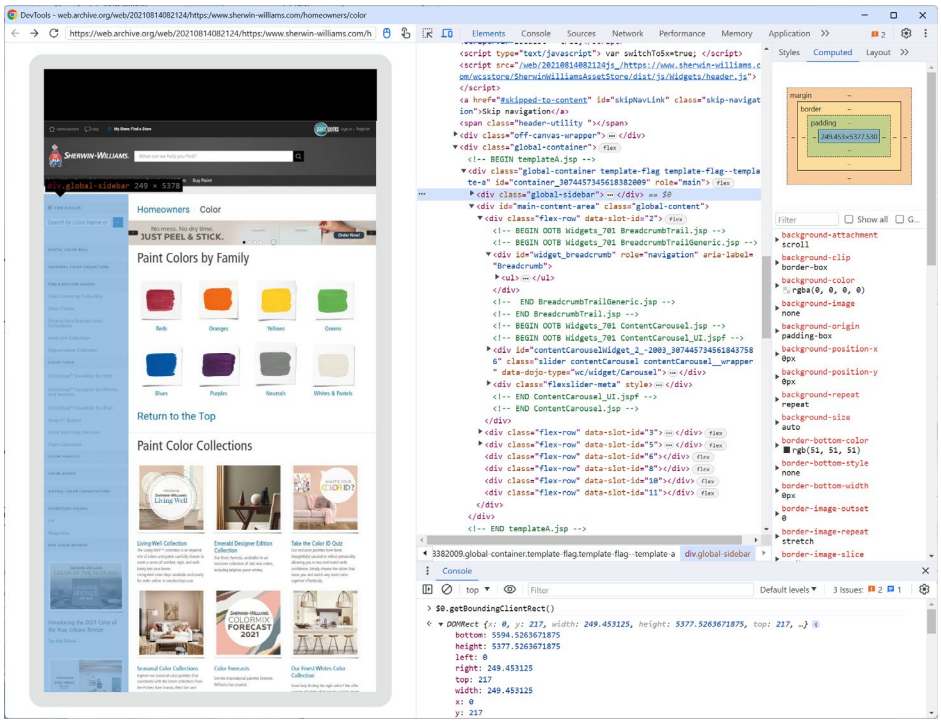
77. The layout, functionality, and design of Web pages are interpreted by a browser (*e.g.*, Samsung Internet, Chrome) "rendering engine," also referred to as a "layout engine" or "browser engine." The rendering engine handles layout and rendering of page content. In addition, the rendering engine handles navigation through hyperlinks and data submitted through forms and implements the Document Object Model (DOM) data structure exposed to page scripts.

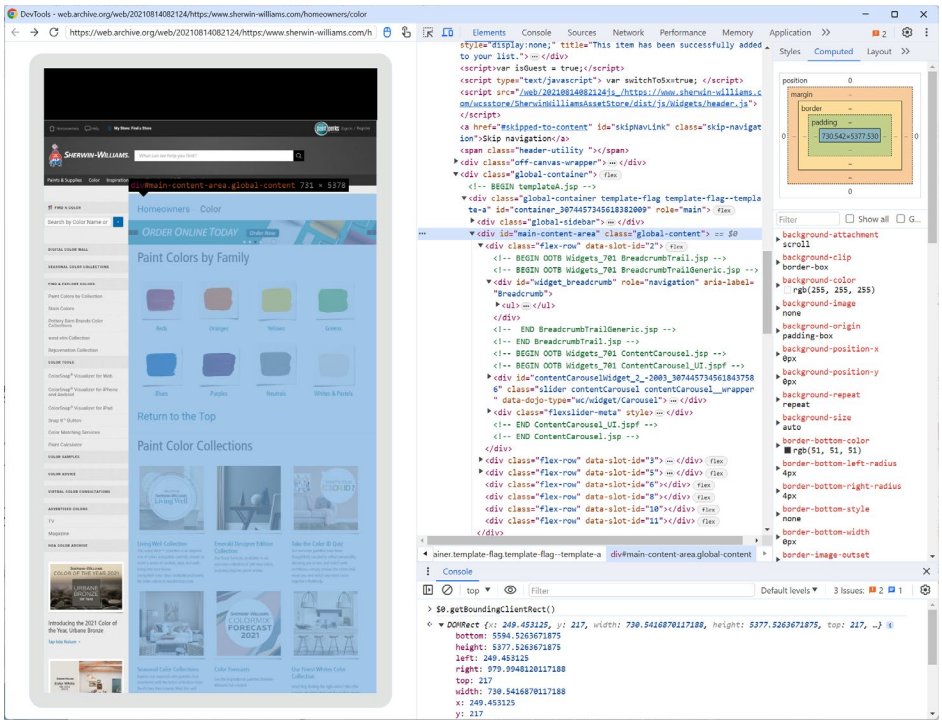
78. The Accused Products enable a user to request a Web page comprising HTML-based Web content including HTML code for at least multiple columns and CSS content. The screenshots below show a user requesting a webpage through the Chrome browser.



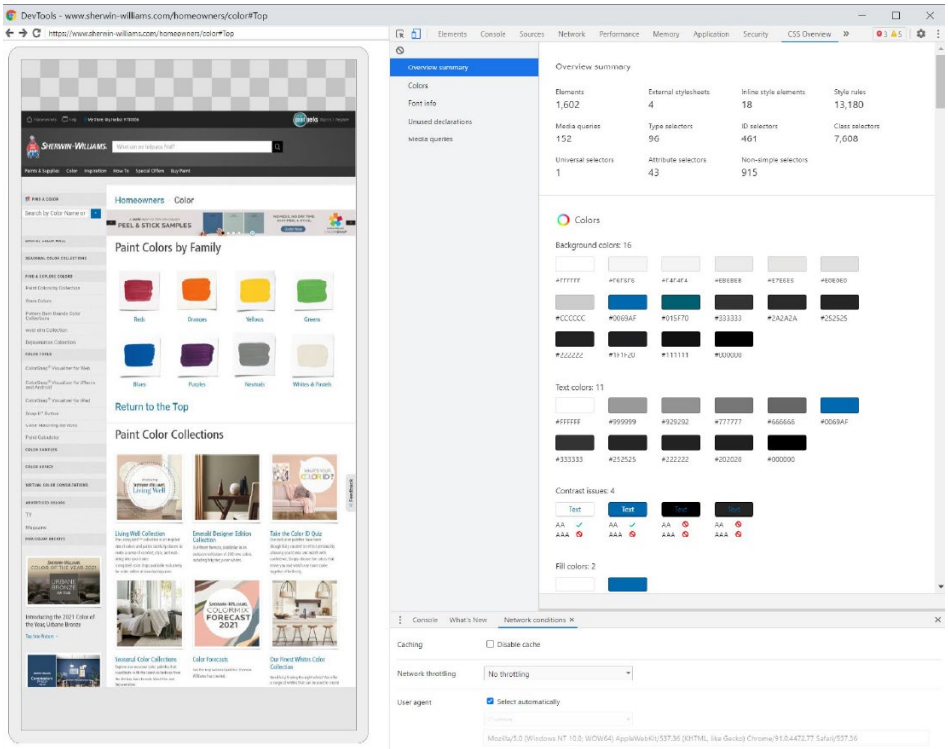
79. The following shows example layout information for a Web page browsed on a Samsung Galaxy S10 using the Chrome Web Inspector running on a laptop connected to the S10

via a USB-C cable. This example shows the layout box for the left-hand column called "global_sidebar" is 249 x 6285 LayoutUnits, in accordance with the CSS box model. There is a similar layout box for each HTML element on the page. There are at least two columns on the page as shown below.

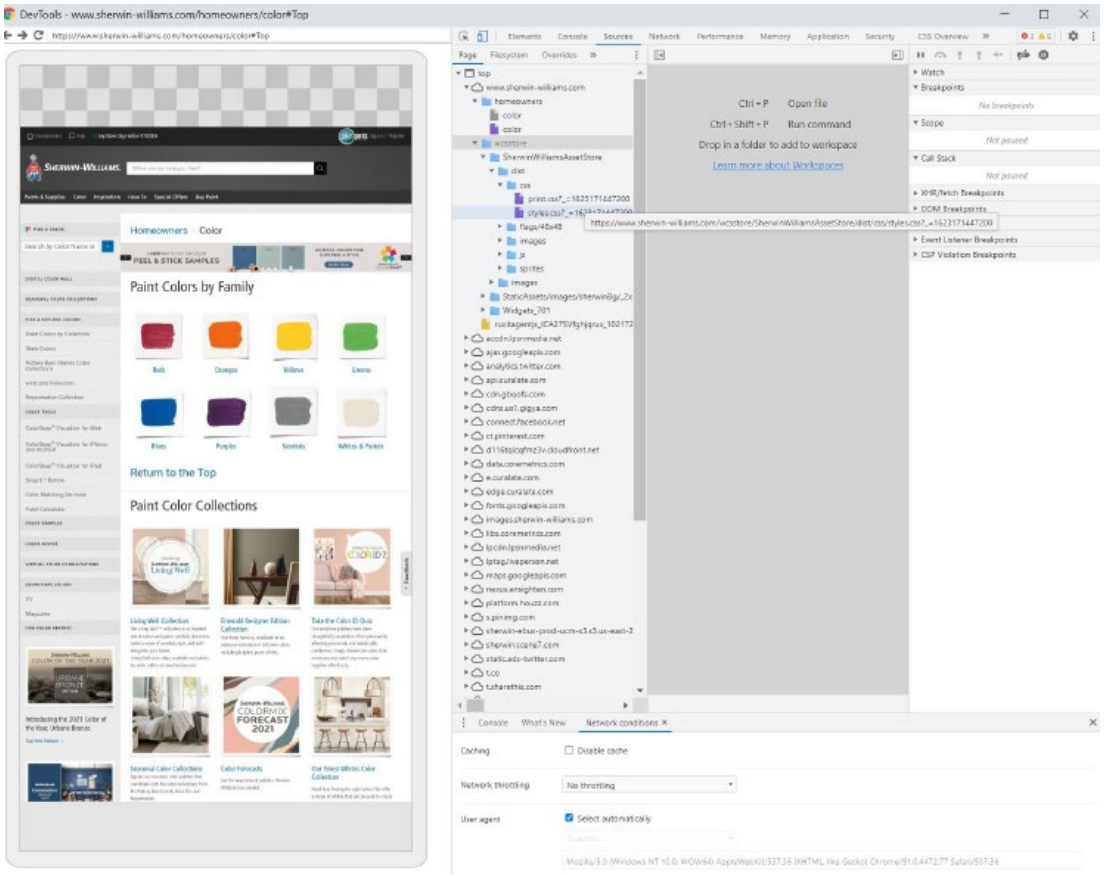




80. The Chrome Web Inspector has a CSS overview tab that provides some information about CSS usage on this page.

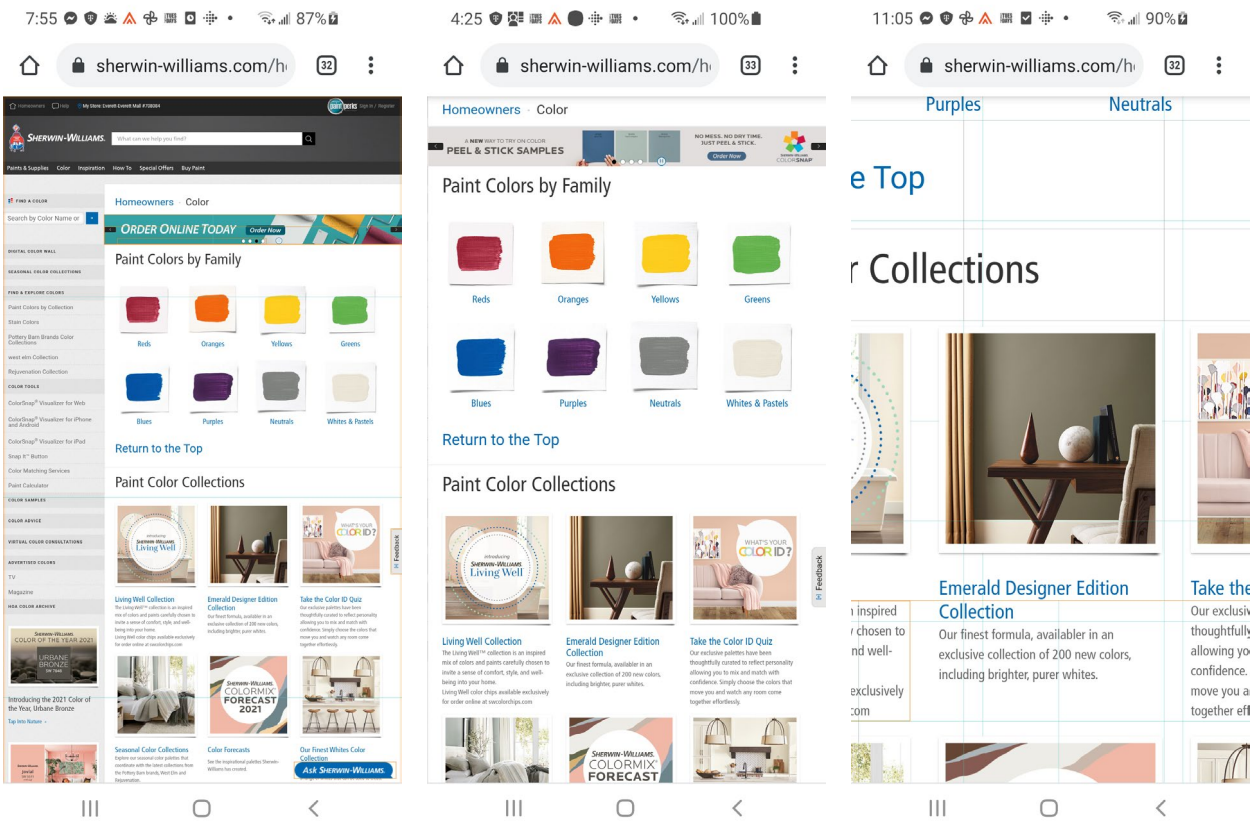


81. The Sources tab of the Chrome Web Inspector shows that two CSS files were used for the Web page, including styles.css used for page layout and styling.



82. Claim 14 recites "retrieving and processing the HTML-based Web content including the HTML code for the at least multiple columns and the CSS content at the mobile device to generate a scalable vector representation of the Web page."

83. The Samsung Internet and Chrome browsers of the Accused Products retrieve and process the HTML-based Web content including the HTML code for the multiple columns and the CSS content at the mobile device to generate a scalable vector representation of the Web page. The screenshots below show zooming in on the scalable vector representation.



84. WebView (and the Samsung Internet and Chrome browsers) creates a vector representation of the entire page, which is scalable.

Picture

- Vector representation of the entire page (not just the visible area)
- No need to go back to webkit when:
  - Scrolling
  - Zooming

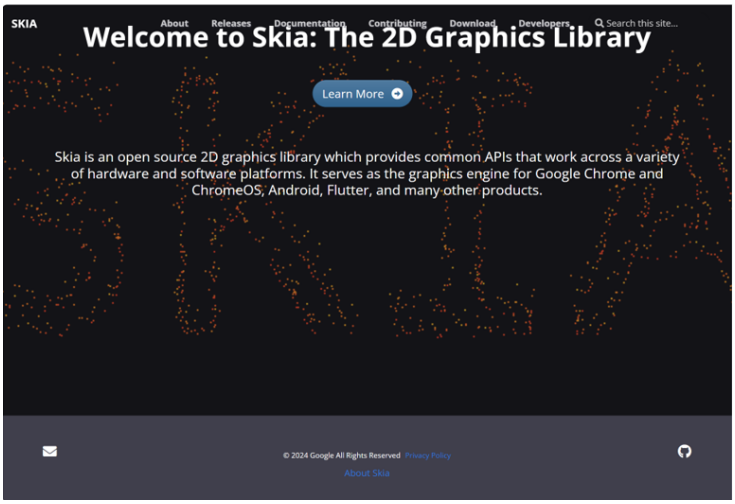
Friday, June 29, 2012

19

19

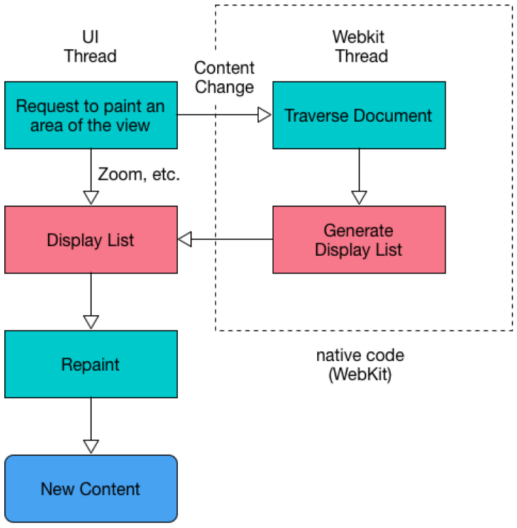
Android WebView — Google IO 2012.

85. The Samsung Internet and Chrome browsers use the Skia 2D vector graphics rendering software/library, which is also referred to as the Skia Graphics Engine.



86. WebView (and the Samsung Internet and Chrome browsers) records lists of painting instructions for the full page and builds display lists. Zooming is accomplished by changing scale and translation parameters in a transformation matrix (SkMatrix) to scale and/or translate position vector coordinates and calling the painting instructions in the display list. While the following describes the approach used by the Chrome browser with WebKit, the current painting process using Blink is similar.

So, instead of directly asking webkit to *paint* the screen, the team had a better idea. They decided to modify Webkit to *record* the list of painting instructions for the full page, but to not paint directly — basically using a "display list" model. The list of painting commands were for example "draw this text here", or "draw this image there".



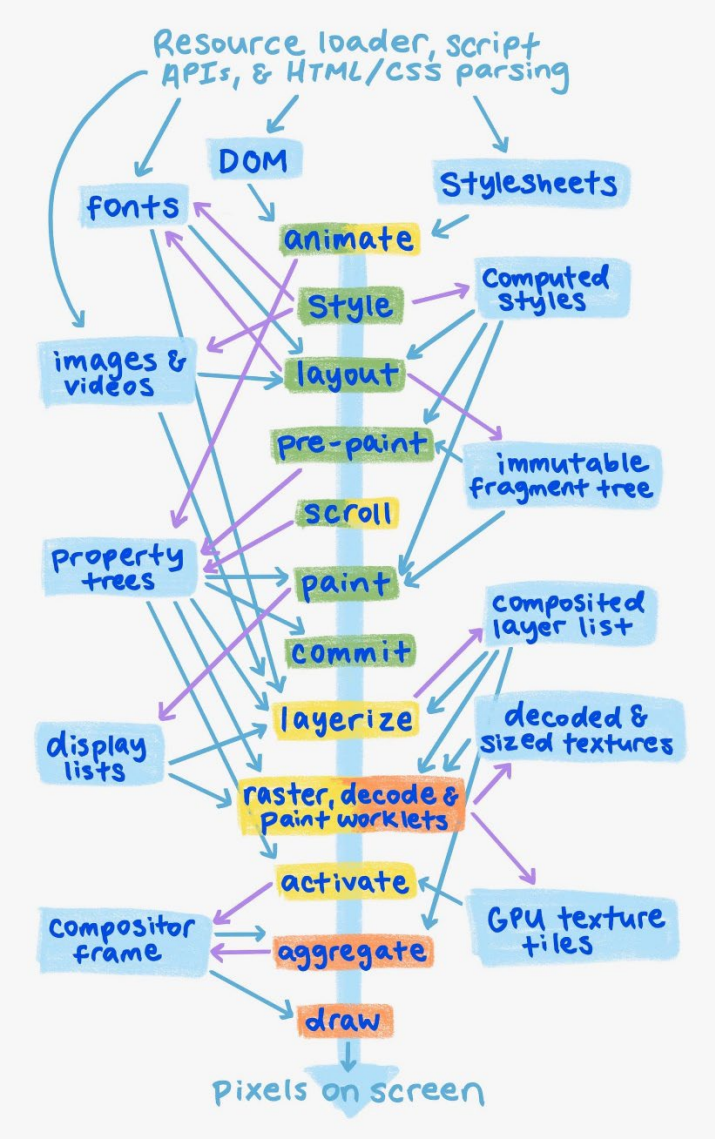
Rendering loop (Android)

When the user would need to zoom in for example, it was not necessary to ask Webkit to repaint the screen — the webview instead could just replay the already present display list, only changing the zoom factor. This also gave a rather nice side-effect that the screen was always sharp — no text blurriness when zooming in, no missing content, as the content was always being redrawn "the right way".

You could think of this display list as a way to have a vector representation of the entire page, that you could scale on demand and redraw — the difference between illustrator and photoshop.

https://medium.com/@camaelon/what-s-in-a-web-browser-83793b51df6c.

87. Display lists are used in the current Rendering Pipeline.


Resource loader, script APIs, & HTML/CSS parsing
DOM
Stylesheets
fonts
animate
Computed styles
Style
images & videos
layout
pre-paint
immutable fragment tree
scroll
property trees
paint
composited layer list
commit
layerize
decoded & sized textures
display lists
raster, decode & paint worklets
activate
GPU texture tiles
compositor frame
aggregate
draw
Pixels on screen

The stages are:

1. *Animate:* change computed styles and mutate *property trees* over time based on declarative timelines.
2. *Style:* apply CSS to the DOM, and create *computed styles*.
3. *Layout:* determine the size and position of DOM elements on the screen, and create the *immutable fragment tree*.
4. *Pre-paint:* compute property trees and invalidate any existing *display lists* and GPU *texture tiles* as appropriate.
5. *Scroll:* update the scroll offset of documents and scrollable DOM elements, by mutating property trees.
6. *Paint:* compute a display list that describes how to raster GPU texture tiles from the DOM.
7. *Commit:* copy property trees and the display list to the compositor thread.
8. *Layerize:* break up the display list into a *composited layer list* for independent rasterization and animation.
9. *Raster, decode and paint worklets:* turn display lists, encoded images, and paint worklet code, respectively, into GPU texture tiles.
10. *Activate:* create a *compositor frame* representing how to draw and position GPU tiles to the screen, together with any visual effects.
11. *Aggregate:* combine compositor frames from all the visible compositor frames into a single, global compositor frame.
12. *Draw:* execute the aggregated compositor frame on the GPU to create pixels on-screen.

https://developer.chrome.com/docs/chromium/renderingng-architecture#pipeline.

Display lists and paint chunks

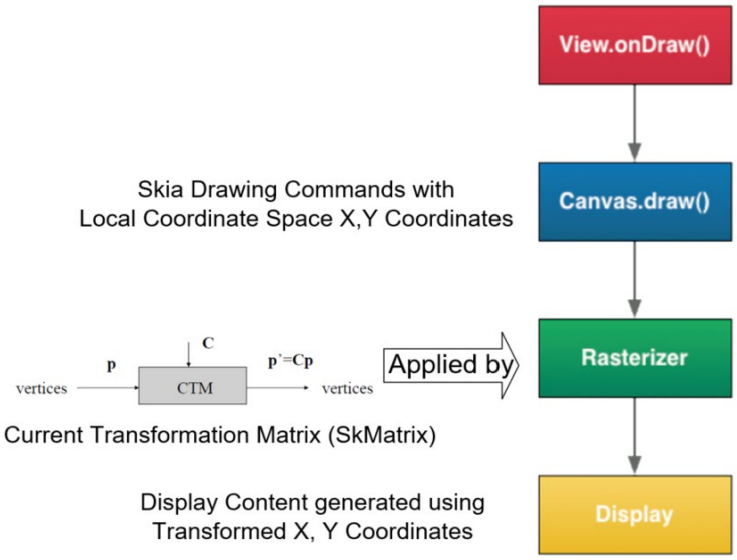
A display item contains low-level drawing commands (see here) that can be rasterized with Skia. Display items are typically simple, with just a few drawing commands, such as drawing a border or background. The paint tree walk iterates over the layout tree and associated fragments following CSS painting order to produce a display item list.

For example:



https://developer.chrome.com/docs/chromium/renderingng-data-structures#display_lists_and_paint_chunks.

88. Skia uses a Current Transformation Matrix (SkMatrix) for scaling and translation of content. An example is shown below for Skia painting.



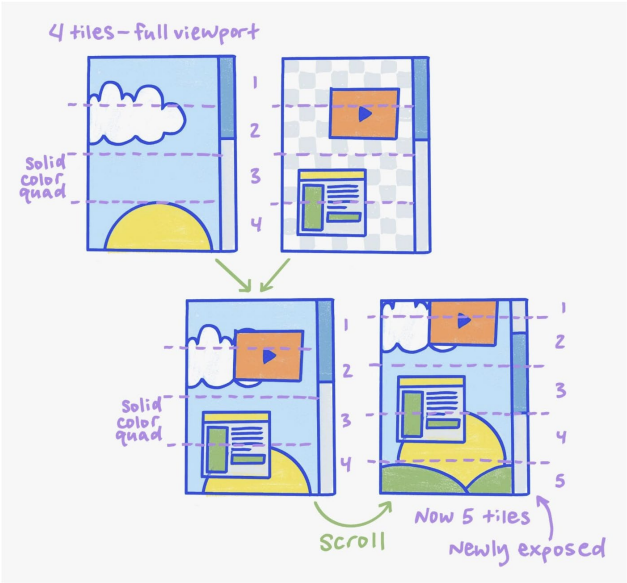
Android Accelerated Rendering Google IO 2011 (modified).

89. The rendering pipeline uses the Skia display lists to rasterize pixels onto GPU texture tiles.

Tiles

In theory, a render process or browser process compositor could rasterize pixels into a single texture the full size of the renderer viewport and submit that texture to Viz. To display it, the display compositor would just have to copy the pixels from that single texture into the appropriate position in the frame buffer (for example, the screen). However, if that compositor wanted to update even a single pixel it would need to re-rasterize the full viewport and submit a new texture to Viz.

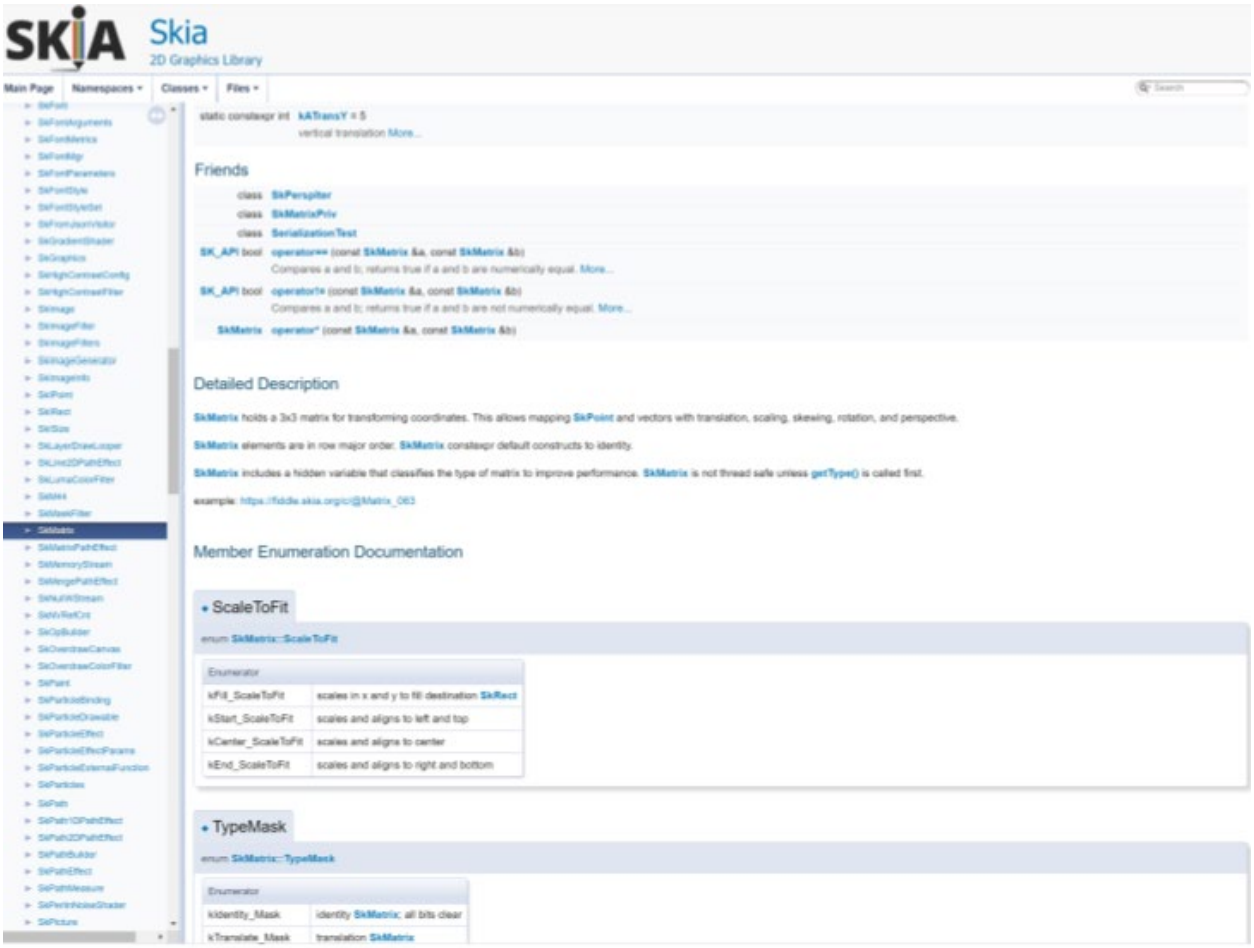
Instead, the viewport is divided into tiles. A separate GPU texture tile backs each tile with the rasterized pixels for part of the viewport. The renderer can then update individual tiles or even just change the position on screen for the existing tiles. For example, when scrolling a website, the position of existing tiles would shift up and only occasionally a new tile would need to be rasterized for content further down the page.



This image depicts an image of a sunny day, with four tiles. When a scroll occurs, a fifth tile begins to appear. One of the tiles happens to only have one color (sky blue), and there is a video and an iframe on top.

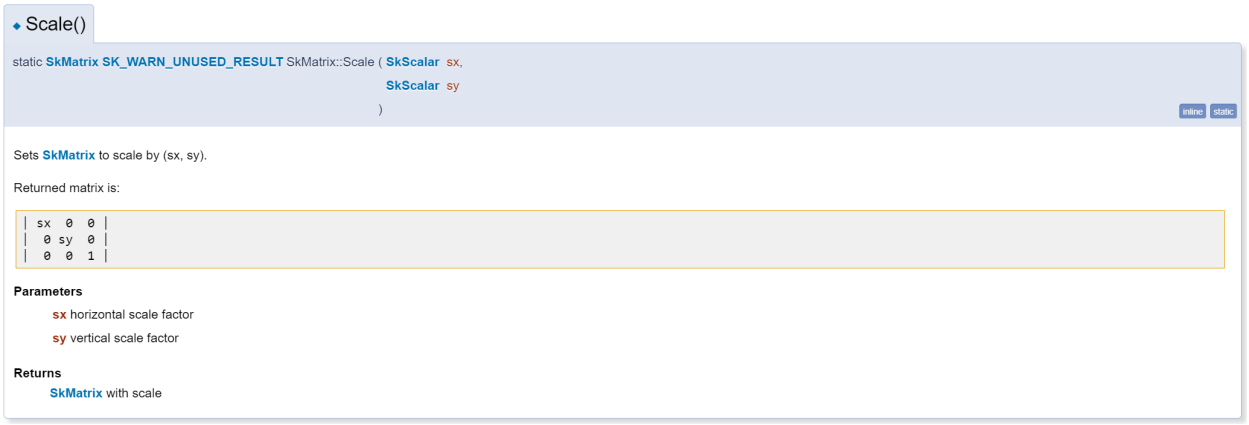
https://developer.chrome.com/docs/chromium/renderingng-data-structures#tiles.

90. The Display List uses the SKMatrix class to effect scaling. SKMatrix holds a 3x3 matrix for transforming coordinates (also known as an affine transformation). SKMatrix is used to effect translation and scaling. Skia renders images by ScaleToFit the destination rectangle (SKRect). SKRect is scaled using the current SKMatrix, and then the Image is scaled to fit the SKRect that is generated.



https://api.skia.org/classSkMatrix.html.

91. The Skia Scale() function can be used to change the sx (horizontal) and sy (vertical) scale factors in the SKMatrix. The setAffine() function can be used to set values for the first two rows of the SKMatrix. This enables scaling and translation to be performed together.



◆ Scale()

static SkMatrix SK_WARN_UNUSED_RESULT SkMatrix::Scale (SkScalar sx,
SkScalar sy
)

inline static

Sets SkMatrix to scale by (sx, sy).

Returned matrix is:

```
| sx  0  0 |
|  0 sy  0 |
|  0  0  1 |
```

Parameters

sx horizontal scale factor

sy vertical scale factor

Returns

SkMatrix with scale



◆ setAffine()

SkMatrix& SkMatrix::setAffine (const SkScalar affine[6])

Sets SkMatrix to affine values, passed in column major order.

Given affine, column, then row, as:

```
| scale-x  skew-x translate-x |
|  skew-y scale-y translate-y |
```

SkMatrix is set, row, then column, to:

```
| scale-x  skew-x translate-x |
|  skew-y scale-y translate-y |
|       0       0           1 |
```

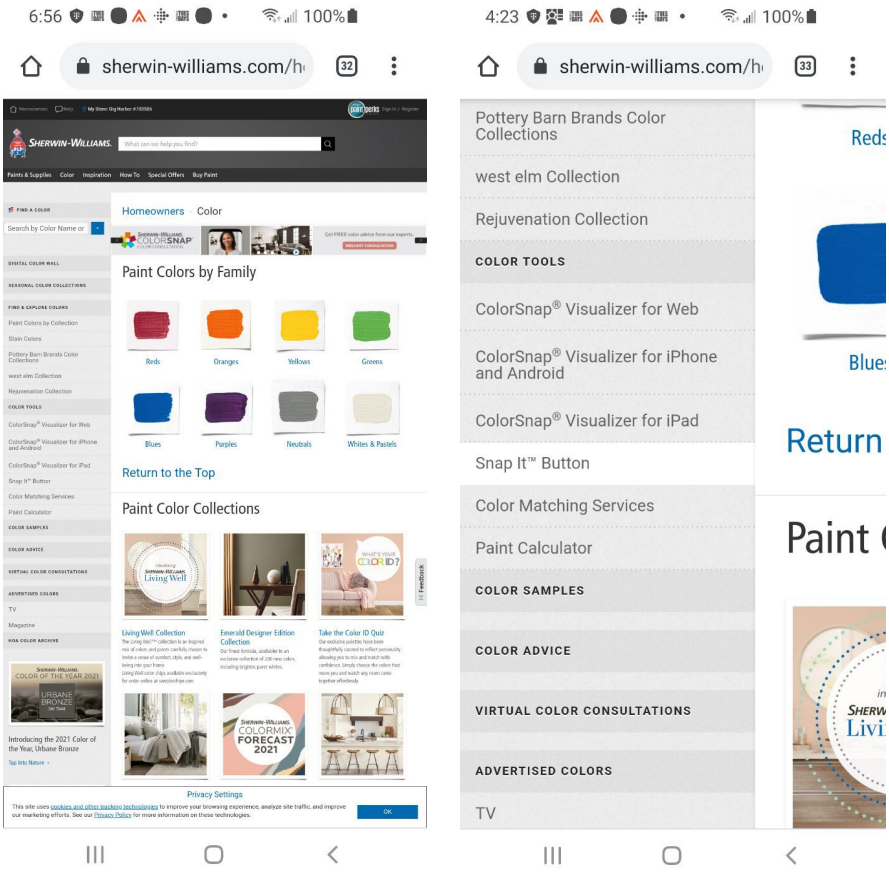
Parameters

affine 3 by 2 affine matrix

92. Claim 14 recites "rendering the Web page on the touch-sensitive display utilizing the scalable vector representation to produce a first view of the Web page at a first zoom level that preserves the original page layout, functionality and design defined by the HTML-based Web content including the at least multiple columns and the CSS content relative to the original page layout, functionality and design defined by the HTML-based Web content including the at least multiple columns and the CSS content when rendered by a commercially available desktop browser; and re-rendering the Web page utilizing the scalable vector representation in response to associated user inputs to iteratively zoom in and out views of the Web page at multiple different zoom levels while preserving the original page layout, functionality, and design defined by the HTML-based Web content including the multiple columns and the CSS content relative to the original page layout, functionality and design defined by the HTML-based Web content including

the at least multiple columns and the CSS content when rendered by the commercially available desktop browser." The Accused Products perform these operations.

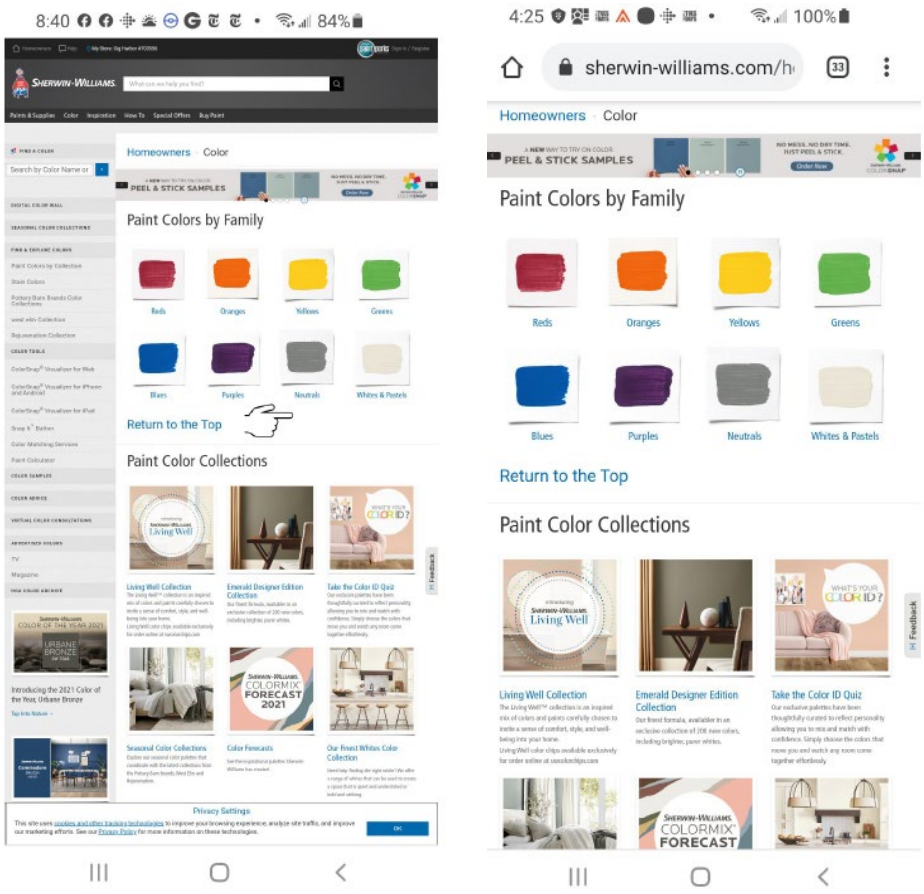
93. As shown by the images below, the Samsung Internet and Chrome browsers of the Accused Products allow users to iteratively zoom in and out of views of a Web Page while preserving the original page layout, functionality, and design defined by the HTML-based Web content including the at least multiple columns and the CSS content when rendered by a commercially available desktop browser.



94. Claim 14 recites "in response to a tap on a first of the multiple columns via the touch-sensitive display, re-rendering the Web page on the touch-sensitive display such that the first of the multiple columns is displayed to fit a width of a display area of the touch-sensitive

display, the display area used for displaying the HTML-based Web content." The Accused Products perform this operation.

95. As shown below, double-tapping on the right-hand column of the Web page using a touch-sensitive display of the Accused Products in the Samsung Internet or Chrome browsers causes the column to fit a width of the display area.



96. Claim 21 of the '729 patent recites:

> 21. The method of claim 14, wherein processing the HTML-based Web content comprises:
> parsing HTML-based code corresponding to the Web page to determine an original page layout of the HTML-based Web content on the Web page;
> logically grouping selected content into objects;
> defining a primary datum corresponding to the original page layout; and, for each object,
> defining an object datum corresponding to a layout location datum for the object's associated display content; and

generating a vector from the primary datum to the object datum for the object.

97. Claim 21 of the ’729 patent recites “The method of claim 14.” *See* above.

98. Claim 21 of the ’729 patent recites “wherein processing the HTML-based Web content comprises: parsing HTML-based code corresponding to the Web page to determine an original page layout of the HTML-based Web content on the Web page; logically grouping selected content into objects; defining a primary datum corresponding to the original page layout; and, for each object, defining an object datum corresponding to a layout location datum for the object's associated display content; and generating a vector from the primary datum to the object datum for the object.” The Accused Products perform these operations.

99. Claim 21 recites “parsing HTML-based code corresponding to the Web page to determine an original page layout of the HTML-based Web content on the Web page.”

100. LayoutNG generates a Layout Tree comprising NGFragment and NGText Nodes.

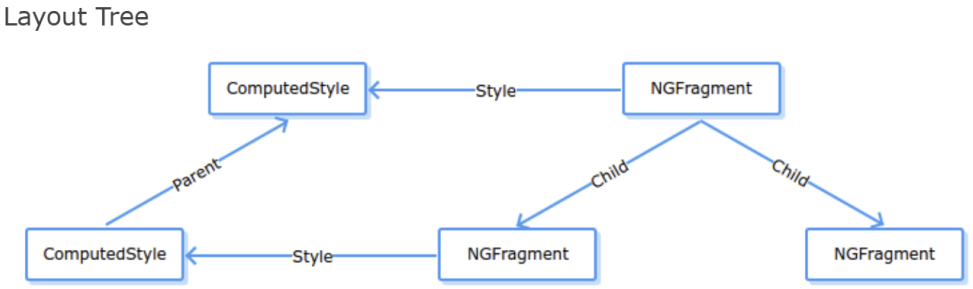


Figure shows the style tree (left) and layout tree (right) and their relationship.

The layout tree is made up entirely of NGFragment and NGText nodes and is immutable. Each node includes a back pointer to the computed style, a list of children and the size and location of the node itself. The location is relative to the immediate parent.

Given the immutability of the tree, use of relative coordinates, and the lack of a parent pointer entire subtrees may be reused across layouts. This concept is discussed in detail in the subsequent layouts section.

101. The Layout Tree also represents the bounding box hierarchy of the HTML document.

Pipelining style, layout, and pre-paint

Collectively, the rendering phases before *paint* are responsible for the following:

- Running the *style cascade* algorithm to calculate final style properties for DOM nodes.
- Generating the layout tree representing the box hierarchy of the document.
- Determining size and position information for all boxes.
- Rounding or snapping sub-pixel geometry to whole pixel boundaries for painting.
- Determining the properties of composited layers (affine transformation, filters, opacity, or anything else that can be GPU accelerated).
- Determining what content has changed since the previous paint phase, and needs to be painted or repainted (paint invalidation).

https://developer.chrome.com/docs/chromium/blinkng.

102. LayoutNG uses ClientRects to size and position HTML Elements (DOM objects) relative to the viewport.

ClientRect	Size and position of an Element relative to the viewport.

Walking up the Tree

ClientRect calculations and IntersectionObserver need to perform a tree-walk up to the root in order to apply transform/layout information. Transform information is not on the layout tree. Adding layout-parent pointers in the Layout Tree would mean that we lose the immutability property; so instead the parent-layout pointers will be stored on the DOM tree. These operations will begin at the DOM node, ask its NGUnplacedFragment for layout information, then walk up the DOM tree to its parent-layout and so forth.

https://docs.google.com/document/d/1uxbDh4uONFQOiGuiumlJBLGgO4KDWB8ZEkp7Rd47fw4/edit?tab=t.0#heading=h.guvbepjyp0oj

103. Claim 21 of the ’729 patent recites “logically grouping selected content into objects.”

104. The Accused Products logically group content into DOM objects.

105. Claim 21 of the ’729 patent recites “defining a primary datum corresponding to the original page layout.”

106. The primary datum corresponding to the original page layout is the upper-left hand corner of the viewport, which is also the origin of the local coordinate space (local coordinate system). The origin of the local coordinate space has x,y coordinates of 0,0.

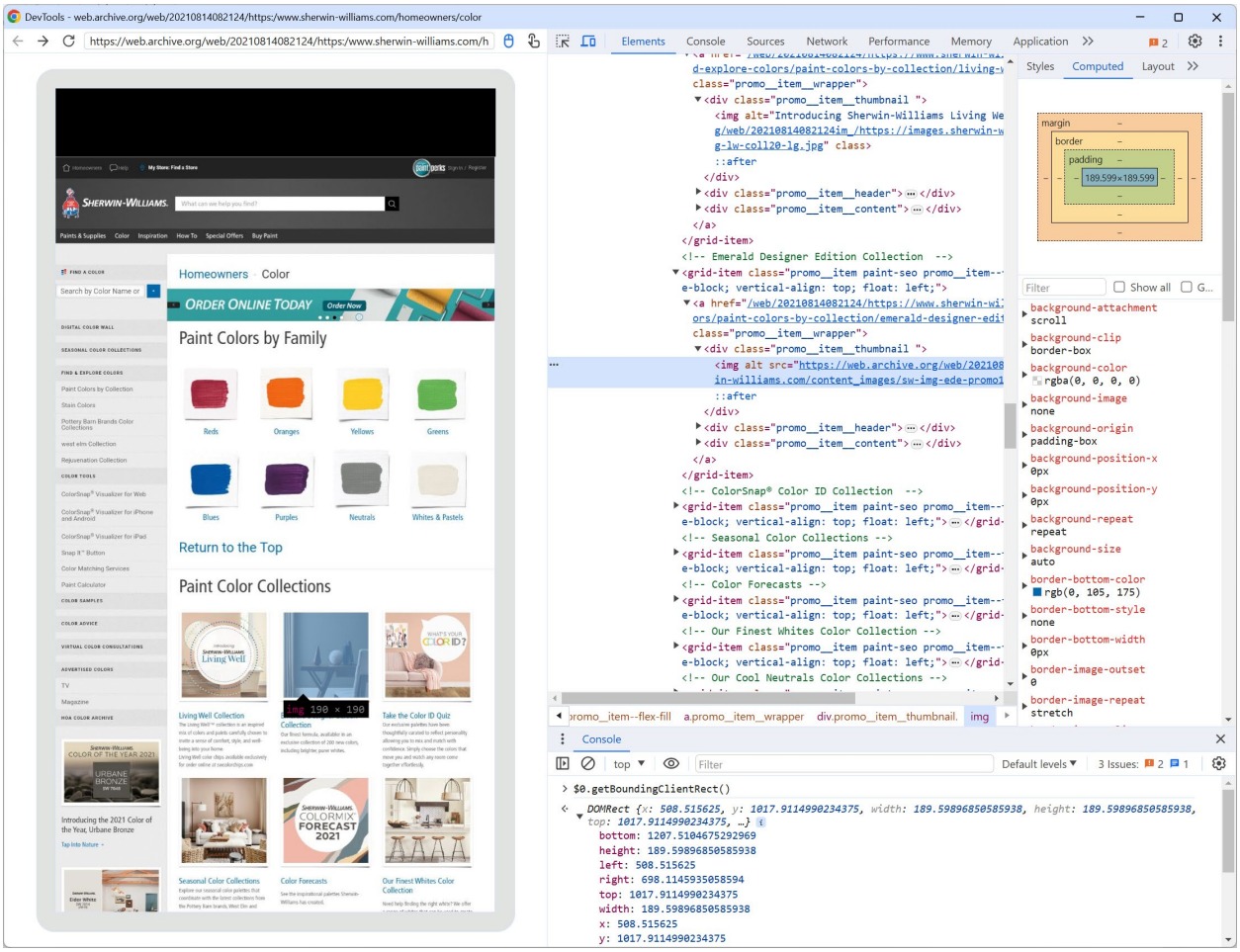
107. Claim 21 of the ’729 patent recites “for each object, defining an object datum corresponding to a layout location datum for the object's associated display content.”

108. Each ClientRect has a defined object datum corresponding to the layout location for the object relative to the viewport origin (0,0). The Blink Rendering engine stores the ClientRect bounding box and DOMRect for each element in the DOM that has associated display content. Each object has an associated x, y coordinate comprising the object datum for the object.

ClientRect	Size and position of an Element relative to the viewport.

https://docs.google.com/document/d/1uxbDh4uONFQOiGuiumlJBLGgO4KDWB8ZEkp7Rd47fw4/edit?tab=t.0#heading=h.guvbepjyp0oj

109. The object datum for the image highlighted in the screenshot below is 508.515 (x), 1017.911 (y) in LayoutUnits.

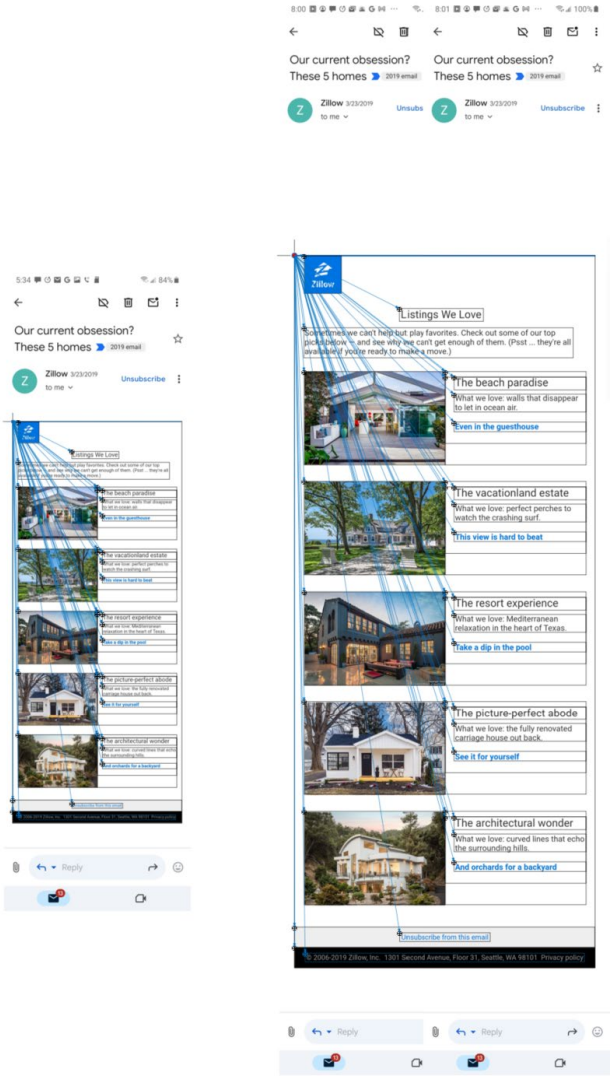


110. Claim 21 of the ’729 patent recites “generating a vector from the primary datum to the object datum for the object.”

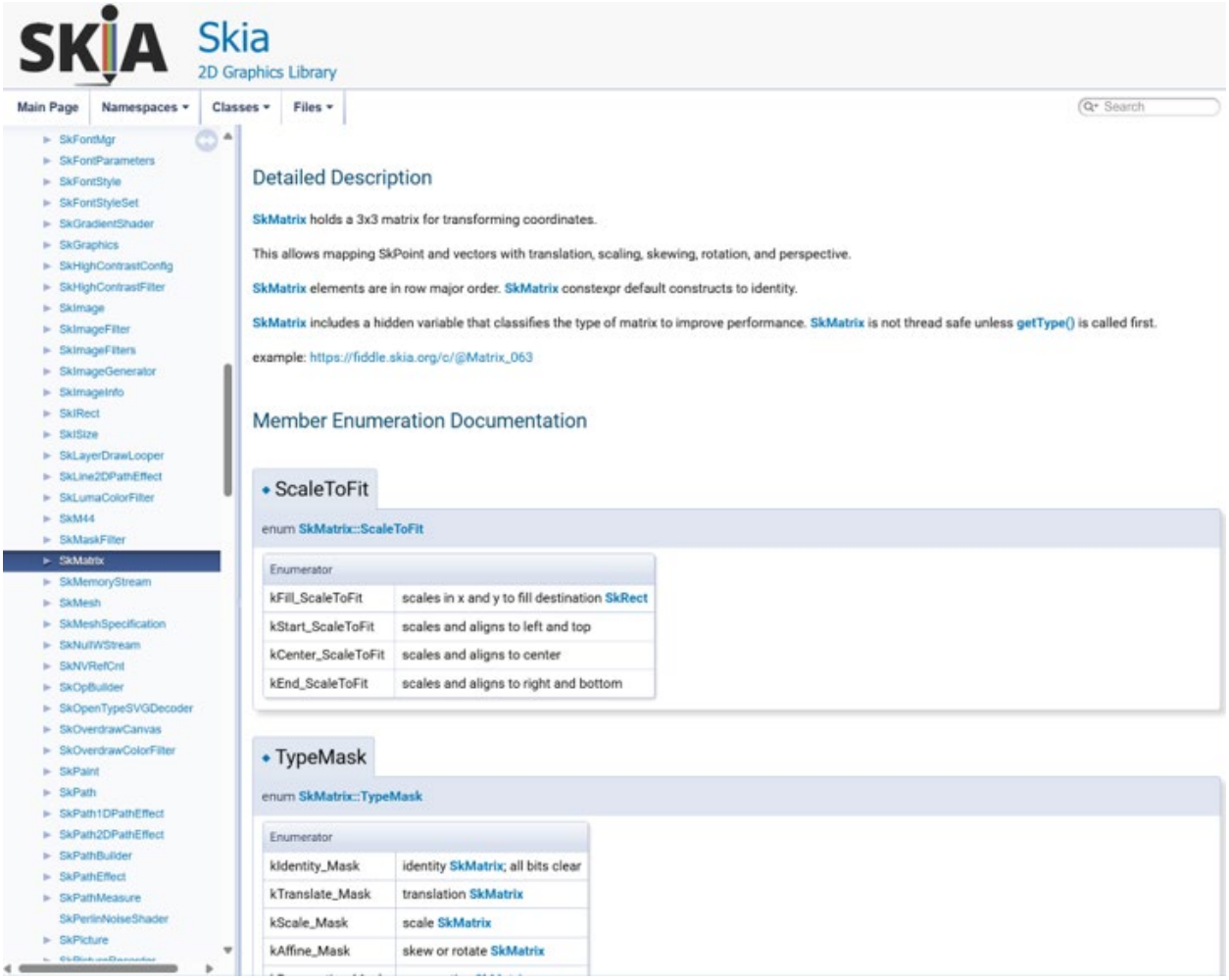
111. The object datums comprise x, y coordinates in the local coordinate space; each x, y coordinate in the local coordinate space is a position vector, with the tail of each position vector at the 0,0 origin (the primary datum) and the head of the position vector at the x, y coordinate (the object datum). The object datums (x, y coordinates) are used to support a vector representation of the Web page that can be scaled and translated to enable the user to zoom in and out views of the Web page with perfect scaling.

112. The following illustrates vector-based scaling of the page layout for a Web page using the Android Chrome browser. This example shows vectors for selected content. The tail of

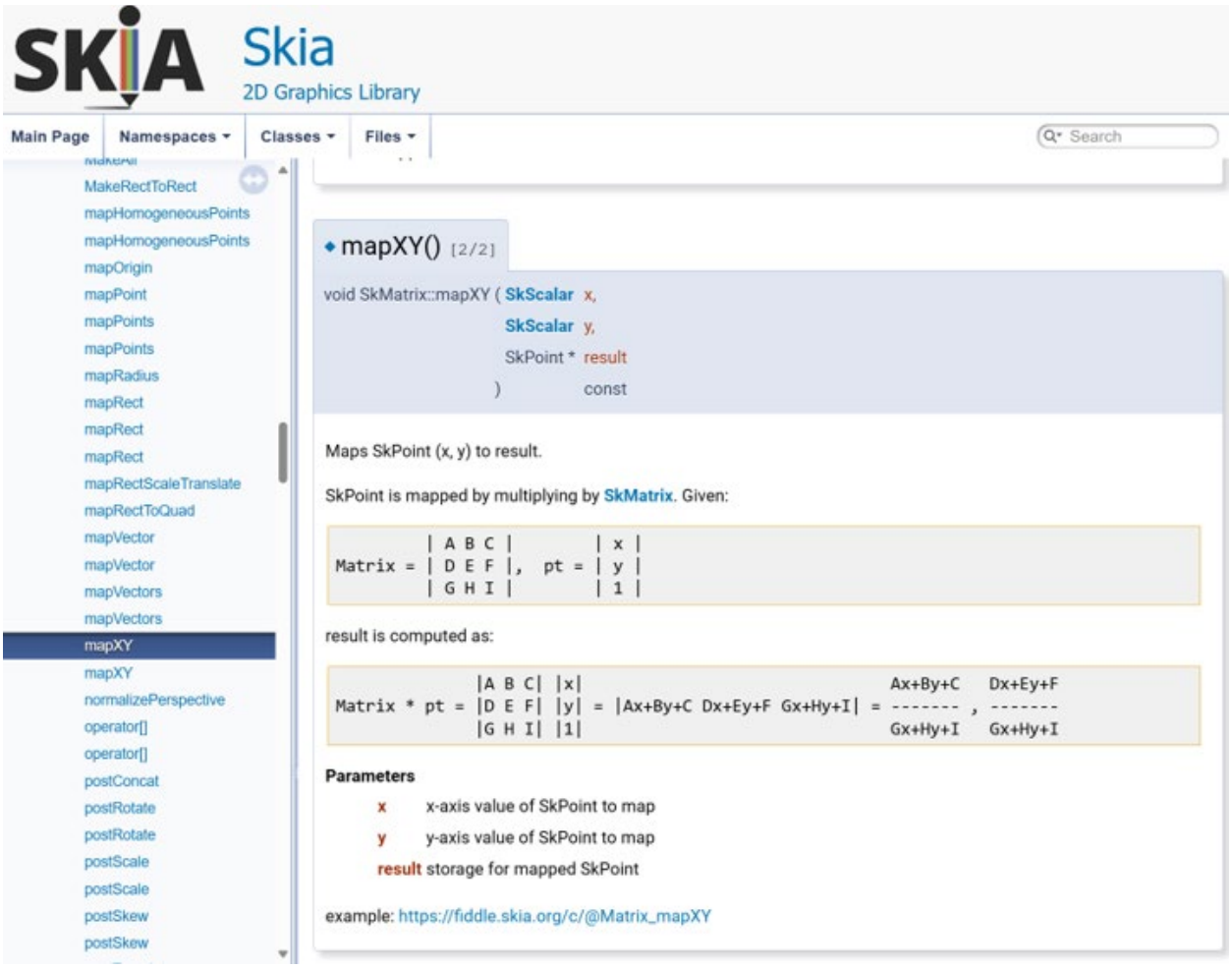
each vector is at the page, viewport, and local coordinate space origin (0, 0) and the head of each vector is at the datum for an associated object.



113. Skia uses vector mathematics to scale and translate the object datums to support zooming and panning (scrolling) and to rasterize tiles. This utilizes the SkMatrix class.



114. Skia provides multiple functions (methods) to scale the coordinates of points (position vectors) in the local coordinate space, such as the mapXY() method. The point is an SkPoint. SkScalars x and y are the x and y scale factors, respectively.



Samsung's Inducement of Infringement of the '729 Patent

115. SoftView incorporates by reference and re-alleges the foregoing paragraphs of this Complaint as if fully set forth herein.

116. The '729 patent was filed on November 8, 2010 as application No. 12/941,106. The '729 patent issued on December 13, 2016. On information and belief, Defendants had knowledge of the '729 patent on or soon after its issuance date.

117. Defendants were served on October 5, 2011, with the Second Amended Complaint in the Delaware litigation alleging infringement of the '353 and '926 patents. The '729 patent is in the same family as those patents. It was pending as application No. 12/941,106 during the Delaware litigation.

118. The ’729 patent also was cited to Defendant SEC during its prosecution of European Patent Application No. EP18847977 in an Annex to the European Search Report dated July 16, 2020. Defendants were thus aware of the ’729 patent at least as of that date.

119. An extended European search report dated July 27, 2020 further cited the ’729 patent as part of a basis for rejecting the EP18847977 application. Defendants were thus aware of the ’729 patent at least as of that date.

120. The application EP18847977 filed October 11, 2020, discussed the ’729 patent. Defendants were thus aware of the ’729 patent at least as of that date.

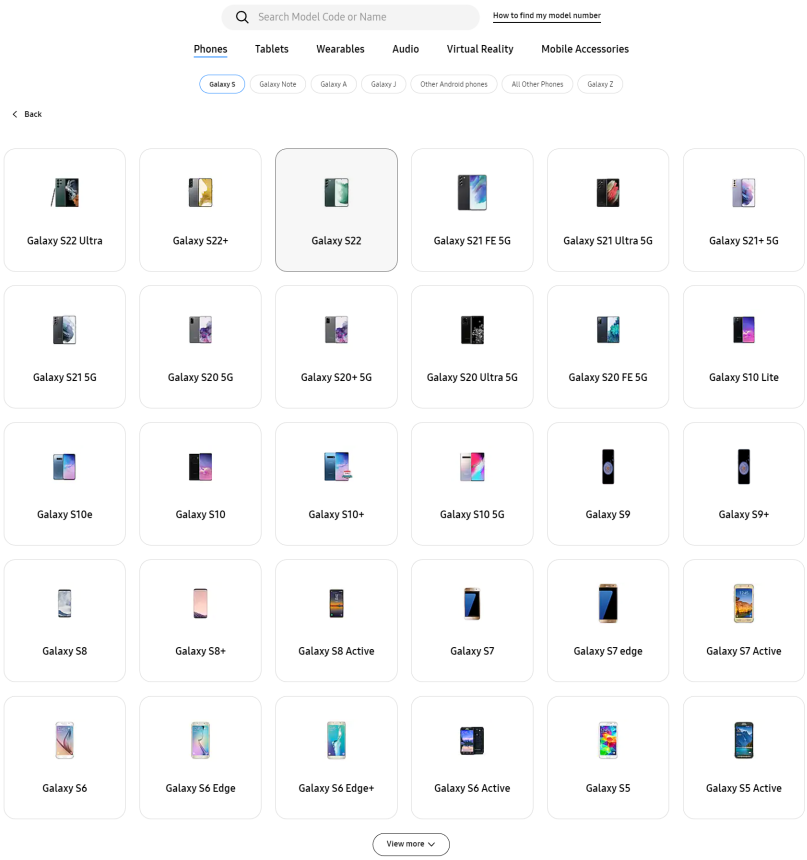
121. On April 11, 2022, during the prosecution of EP18847977, SEC submitted a filing to the European Patent Office discussing the ’729 patent and attempting to argue around it as prior art. Defendants were thus aware of the ’729 patent at least as of that date.

122. SoftView is informed and believes, and thereon alleges, that Defendants have actively induced infringement of at least claim 21 of the ’729 patent in violation of 35 U.S.C. § 271(b) by knowingly and intentionally encouraging or aiding third parties to use infringing software and hardware products in this judicial district and elsewhere throughout the United States, without license or authority from SoftView, including at least the Accused Products. SoftView is informed and believes, and thereon alleges, that these third parties have infringed at least claim 21 of the ’729 patent in violation of 35 U.S.C. § 271(a) by using infringing software and hardware products, including the Accused Products. The Defendants, through at least their user manuals, Web site support and training materials actively induced their customers and users of the Accused Products to infringe the ’729 patent. Furthermore, the natural and intended use of the Accused Products (*e.g.*, browsing the internet with Samsung Internet and Chrome) infringes at least claim

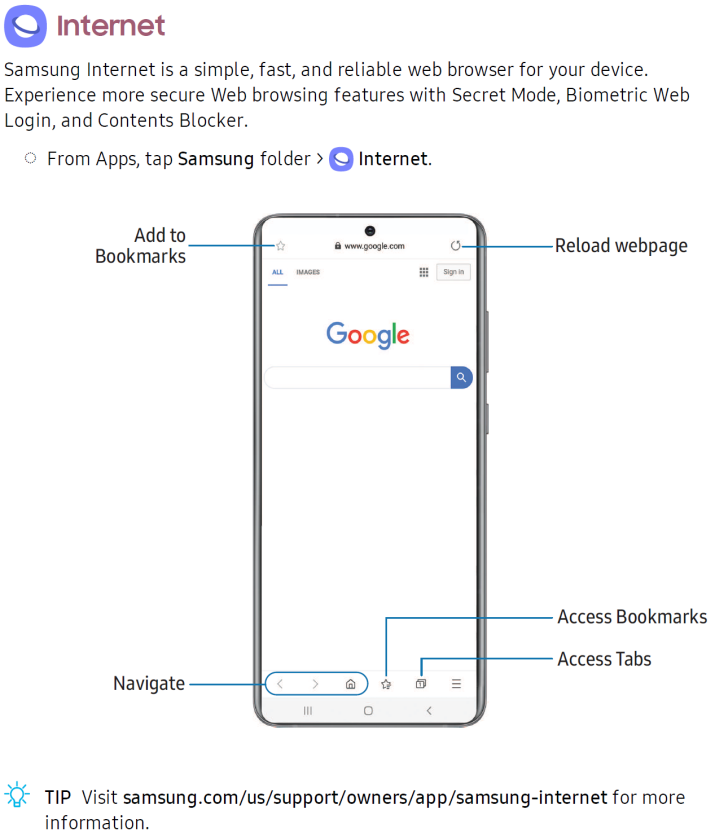
21 of the ’729 patent. Defendants know and intend that the Accused Products are used in this infringing manner.

123. Defendants have also knowingly induced infringement of at least claim 21 in the ’729 patent by providing documents and instructions demonstrating how to use the devices to infringe the claims, and/or by providing service, support, or other active assistance to its customers in using the Accused Products in the U.S. The documentation which Defendants have provided includes, at least: (i) the product information for the Accused Products set forth on Defendants’ websites, including https://www.samsung.com/us/, which includes various manuals, specifications, and other technical documentation for the Accused Products provided on Defendants’ websites; and (ii) the other product documentation which, on information and belief, Defendants provide in electronic and/or paper form to its customers for the Accused Products.

124. For example, the Samsung interactive support page at https://www.samsung.com/us/support/ enables users to access manuals for Samsung products, as illustrated below. Selecting a mobile phone tile will enable the user manual for that mobile phone to be downloaded. Samsung has provided a means for accessing an electronic copy of the user manuals for Samsung mobile devices using this URL since at least 2012.

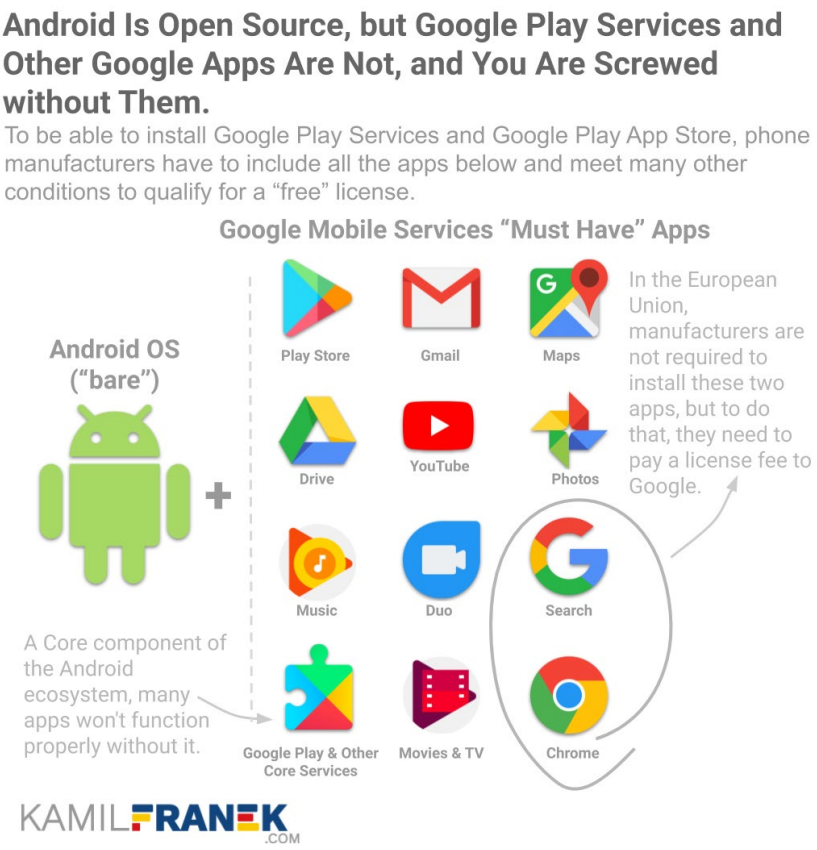


125. Each user manual includes a section covering the Samsung Internet (browser) application that provides graphics and text describing how to use the Samsung Internet browser. For example, the following shows a portion of the first page describing the Samsung Internet browser in the Samsung Galaxy S20 | S20+ | S20 Ultra 5G user manual.

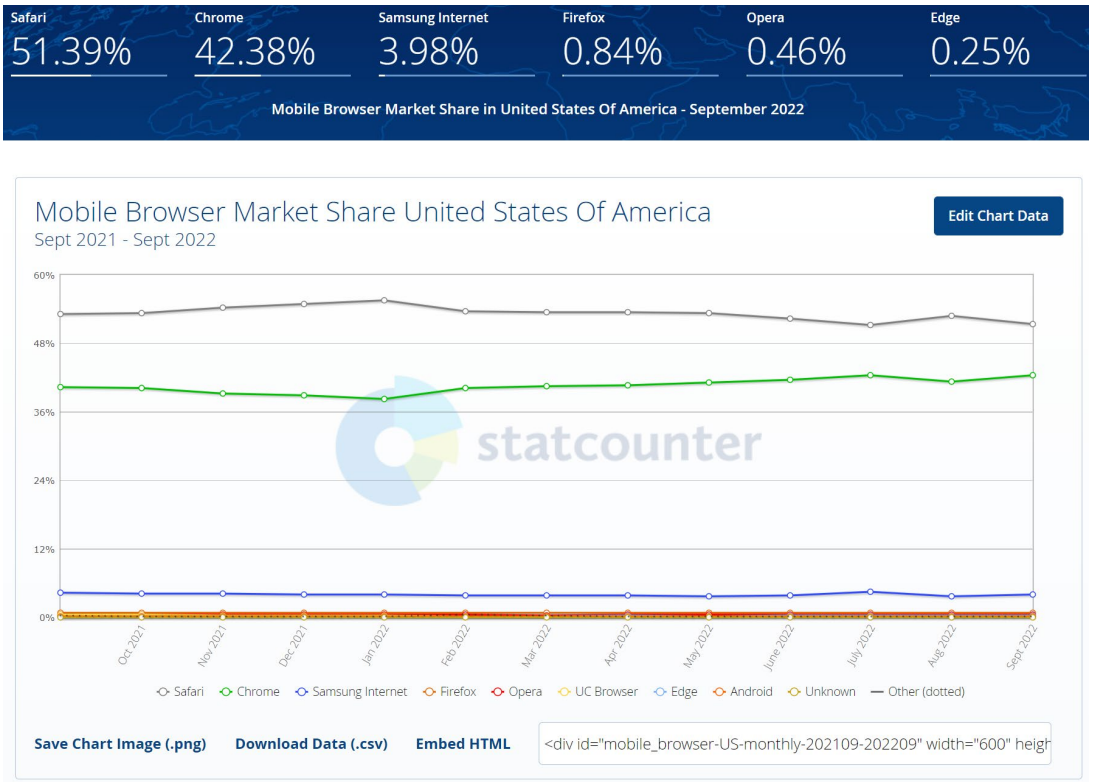


126. In addition to the instructions provided in the user manuals, knowledge by users of how to browse Web pages using a mobile browser such as the Samsung Internet application was ubiquitous throughout the damages period, including use of gestures such as tap-to-zoom that were pioneered by SoftView (SoftSource) and copied by the Samsung Internet and Android Chrome browsers.

127. Samsung smartphones and tablets come with two browsers pre-installed – the Samsung Internet browser and the Android Chrome browser. By contract, the Android Chrome browser is required to be installed on Android devices since it was introduced in 2012 (to access the Google Play Store and install Google Play Services). Samsung could remove it, but then they would have to pay for Android (this is one of Google's criteria for providing Android for "free"). See https://www.kamilfranek.com/how-google-makes-money-from-android/, which includes the following graphic:



128. Android Chrome and the Android Internet browsers have similar user interfaces and share the same "Blink" rendering engine and associated software libraries, frameworks, and APIs. Samsung is fully aware that users of its devices use Android Chrome—usage statistics from services such as Statscounter.com show Android Chrome is used approximately 5-6 times as much as the Android Internet browser on Samsung smartphones and tablets in the United States during the past year. https://gs.statcounter.com/browser-market-share/mobile/united-states-of-america. During the timeframe Samsung had an Android smartphone market share of approximately 60% in the U.S.



129. Samsung has received and continues to receive billions of dollars from Google for pre-configuring the Google search engine as the default search engine for the Samsung Internet browser and the Android Chrome browser. See, *e.g.*, https://www.business-standard.com/article/companies/google-pays-hefty-sums-to-remain-top-search-engine-says-us-justice-dept-122090900155_1.html.

130. Defendants have induced infringement of at least claim 21 of the '729 patent.

131. On information and belief, Defendants have committed acts of infringement notwithstanding actual knowledge of the '729 patent and without a good faith basis to believe that their activities do not infringe at least claim 21 of the '729 patent. All infringement of the '729 patent following Defendants' knowledge of the '729 patent is willful, and SoftView is entitled to treble damages and attorneys' fees and costs incurred in this action under 35 U.S.C. §§ 284 and 285.

132. In the event Defendants assert they were not aware of ’729 patent or lack the requisite knowledge and intent to induce infringement, Defendants would have been willfully blind of SoftView’s patent rights.

133. In view of the Delaware litigation and multiple patent applications pending (including the ’729 patent application) and that issued during the litigation from the same family as the ’353 and ’926 patents, along with the aforementioned discussion of the ’729 patent in Defendants’ patent applications, Defendants were aware of a high probability of the existence of such applications and patents and their infringement of them, and/or would had to have taken deliberate actions to avoid learning of the existence of such applications and patents and their infringement of them, including the ’729 patent.

134. Defendants’ actions are at least objectively reckless as to the risk of infringement and this objective risk was either known or should have been known by Defendants.

135. Defendants’ direct and indirect infringement of the Asserted Patents is and has been willful, intentional, deliberate, and/or in conscious disregard of SoftView’s rights under the Asserted Patents.

136. SoftView has incurred substantial damages, including monetary damages.

137. SoftView has been irreparably harmed by Defendants’ infringement of the ’729 patent.

138. Therefore, SoftView is entitled to actual and/or compensatory damages, reasonable royalties, pre-judgment and post-judgment interest, enhanced damages, and costs.

COUNT II – INFRINGEMENT OF U.S. PATENT NO. 7,844,889

139. SoftView incorporates by reference and re-alleges the foregoing paragraphs of this Complaint as if fully set forth herein.

140. Defendants have directly infringed at least claim 5 of the ’889 Patent.

141. As a matter of law, the notice provisions of 35 U.S.C. § 287 do not apply in this case for the ’889 patent, and therefore SoftView has satisfied its obligations under 35 U.S.C. § 287 with respect to the ’889 patent.

142. At the time of the claimed invention, reproducing web pages on a variety of devices presented a significant technical problem. Most internet content displayed as a flat single resolution with no browser support for zoom. ’353 Patent, 2:1-3. Much of the content on the internet was designed for display on desktop computers with a single target resolution (e.g., 640x480, 800x600). ’353 Patent, 2:4-5. Major internet content providers chose to create their Web pages using fixed resolution structures, such as tables, which gave them the ability to control the look and feel of their Web sites. ’353 Patent, 2:5-10. The fixed resolution approach was so ubiquitous that it became the most common method to brand or uniquely identify websites. ’353 Patent, 2:10-13.

143. While the fixed resolution approach was useful for site branding and product differentiation, it presented a daunting technical problem: how to display Internet content designed for desktop computers on devices with smaller screens, lower resolution, and/or different aspect ratios, such as cell phones and handheld computers. ’353 Patent, 2:13-18. For example, a website designed for the fixed resolution of a larger desktop display would not fit correctly on small screens, low resolution screens, or devices with different aspect ratios.

144. The ’889 patent claimed invention provides a solution to this daunting technical problem. Claim 5 is a dependent claim that (together with claim 1) recites as follows:

> **1.** A method comprising:
> in response to a request to access a Web page made via a wireless hand-held device including a display,
> retrieving markup language-based Web content associated with the Web page having an original format defining an original page layout of content on the Web

page, the content including text content, a plurality of hyperlinks, and a plurality of images;
processing, via the wireless hand-held device, the markup language-based Web content with a rendering engine to generate page layout information corresponding to the original page layout of content on the Web page as interpreted by the rendering engine including a width of the Web page;
employing the page layout information to generate scalable page layout information; and
employing scalable page layout information and/or data derived therefrom to generate views of the Web page on the display, including,
a view under which the width of the Web page is fully displayed; and
views of the Web page generated in response to associated user inputs to enable a user to iteratively zoom in on a portion of the Web page while preserving the original page layout, functionality, and design of the content on the Web page defined by the markup language-based Web content,
wherein preservation of the functionality defined by the markup language-based content includes preservation of hyperlink functionality.

5. The method of claim 1, further comprising:
parsing and processing markup language code with the rendering engine to determine the original page layout of display content within the Web page as interpreted by the rendering engine, wherein the original page layout defines a layout location for a plurality of objects, including text objects associated with the text content and image objects associated with the plurality of images included in the Web page; and
for each object,
defining an object datum corresponding to the layout location for the object on the original page layout; and
associating the object and its object datum.

145. As claimed in claim 1 of the ’889 patent, HTML-based Web content is translated into scalable page layout information, which is then used to generate views of the Web page on a display. At the time of the claimed invention, this process was not well-known, routine, or conventional. The Web page can be iteratively zoomed in on by a user while preserving the original page layout, functionality, and design of the content on the Web page defined by the markup language-based Web content. Prior art solutions did not provide this novel iterative zooming functionality that supports features we take for granted today, such as pinch zooming on a smartphone or tablet, and it was not well-known routine, or conventional.

146. Claim 5 of the ’889 patent further specifies operations that include defining an object datum corresponding to the layout location for the object on the original page layout and associating the object and its object datum. At the time of the claimed invention, this process was not well-known, routine, or conventional. This object datum permits vectors to be defined between the object and a primary datum, thereby allowing a vectorized view of the Web page. An example is shown in the exemplary Figure 4C of the patent specification, where the primary datum is in the upper left corner of the screen and each arrow points to an object datum.

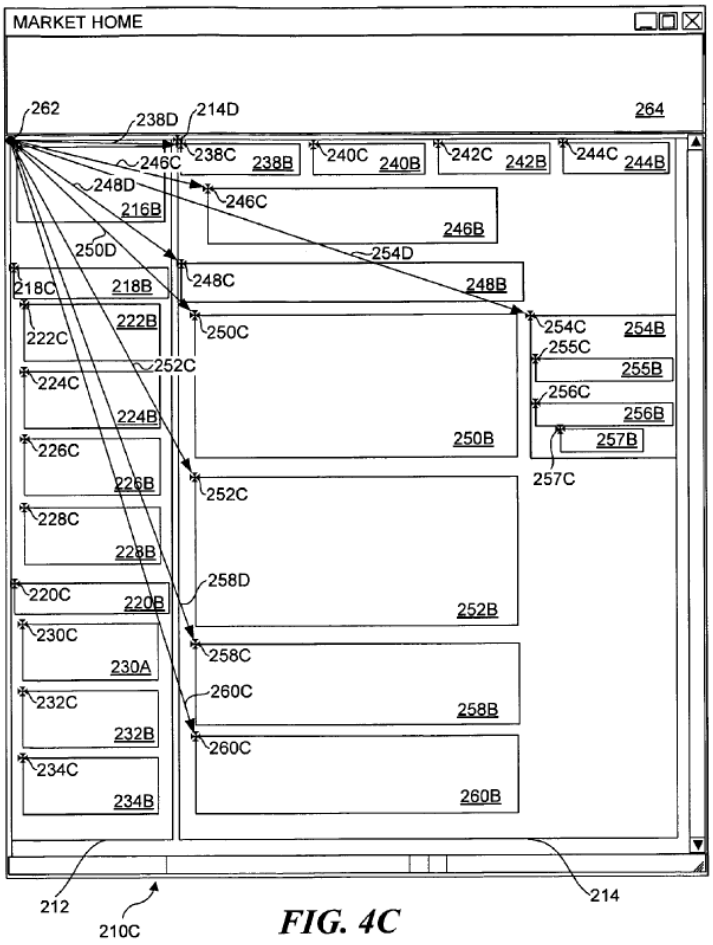


FIG. 4C

147. Claim 5 of the ’889 patent is directed to a technological solution to a technological problem, and was not well-known, routine, or conventional.

148. The advantages of the '889 patent's claimed technical solution are substantial. Web pages could now "be rapidly rendered, zoomed, and panned. Moreover, the rendered displays provide substantially the same or identical layout as the original Web page, enabling users to easily navigate to selected content and features on familiar Web pages." '353 Patent, Abstract. This meant that users could finally be presented with web pages that fit and/or were effectively usable on devices with smaller screens, different resolutions, or different aspect ratios.

149. With the claimed invention, the web page could be presented as it would be on a desktop computer (as most web pages were designed for), and then rapidly panned and zoomed to facilitate use on devices with smaller screens, different resolutions, or different aspect ratios. A Web page that was unusable when presented with the fixed resolution approach could now be zoomed to suit the device display. Today, this functionality is pervasive on smartphone and tablet devices and taken for granted. But prior to the claimed invention, it did not exist.

150. In addition, the claimed invention allowed the look and feel of the Web page to be maintained, even when designed for fixed resolutions. This allowed content providers to continue branding their Web pages with fixed resolution designs. It also allowed users to access Web pages as they were used to seeing them on desktop computers suited to their fixed resolution design, which meant users knew where to locate items on the page based on its familiar look and feel. Thus, the claimed invention made web pages more convenient for users.

151. The claimed invention has an additional advantage: internet content providers can rely on the fixed resolution approach instead of designing numerous different versions of the same web page to suit the screen size, resolution, and aspect ratio of different devices. For example, a web page designed for a 16-inch desktop display could be effectively viewed and used on a small hand-held mobile phone. This is superior to the alternative approach of designing multiple

versions of the same web page to suit different devices, which was time consuming, costly, and impractical for many Web content developers.

152. Defendants have directly infringed (literally and/or under the doctrine of equivalents) at least claim 5 of the '889 patent in violation of 35 U.S.C. § 271 *et seq.*, through use, including testing of the Accused products without authority or license of the Accused Products.

153. Claim 5 of the '889 patent depends from claim 1.

154. Claim 1 recites "a method comprising." To the extent the preamble is limiting, the Accused Products perform a method as further described in the following paragraphs.

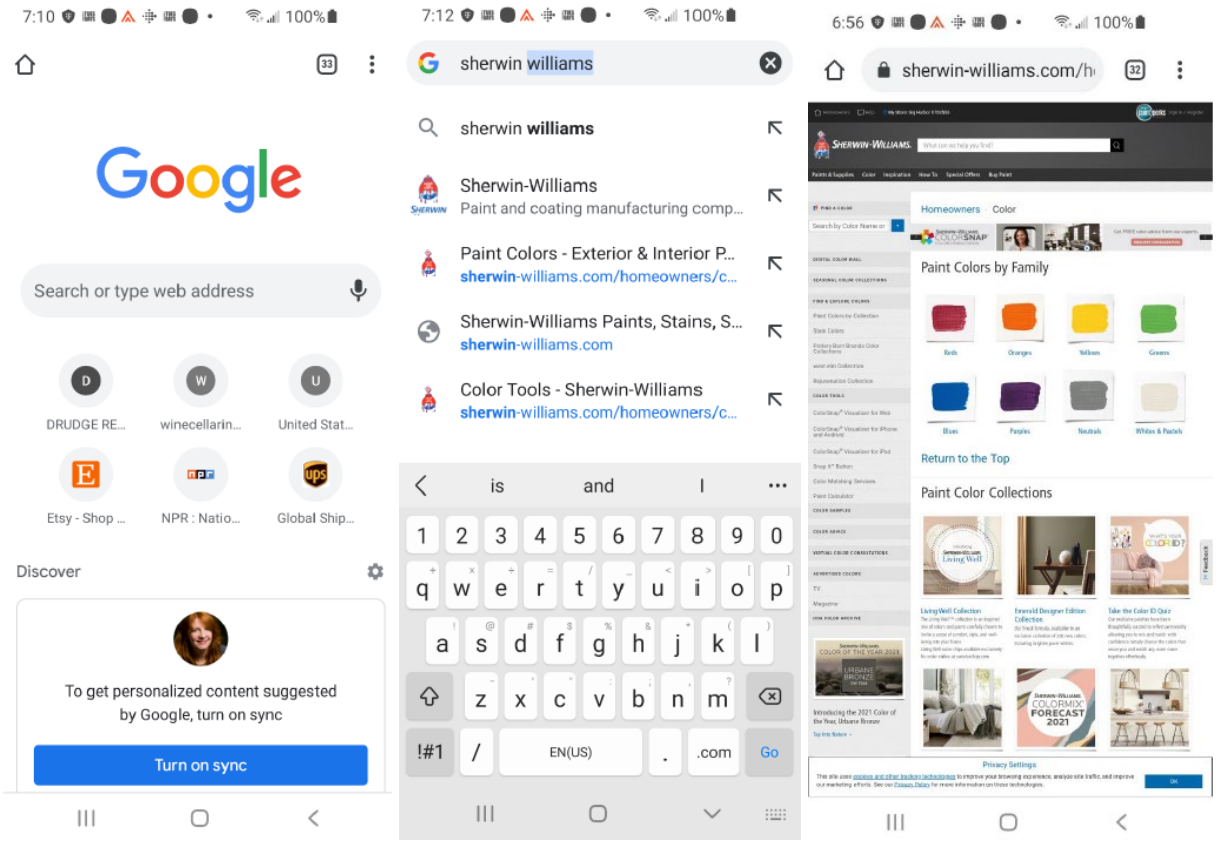
155. Claim 1 of the '889 patent recites: "in response to a request to access a Web page made via a wireless hand-held device including a display, retrieving markup language-based Web content associated with the Web page having an original format defining an original page layout of content on the Web page, the content including text content, a plurality of hyperlinks, and a plurality of images." The Accused Products perform this operation.

156. The Accused Products are wireless hand-held devices including a display.



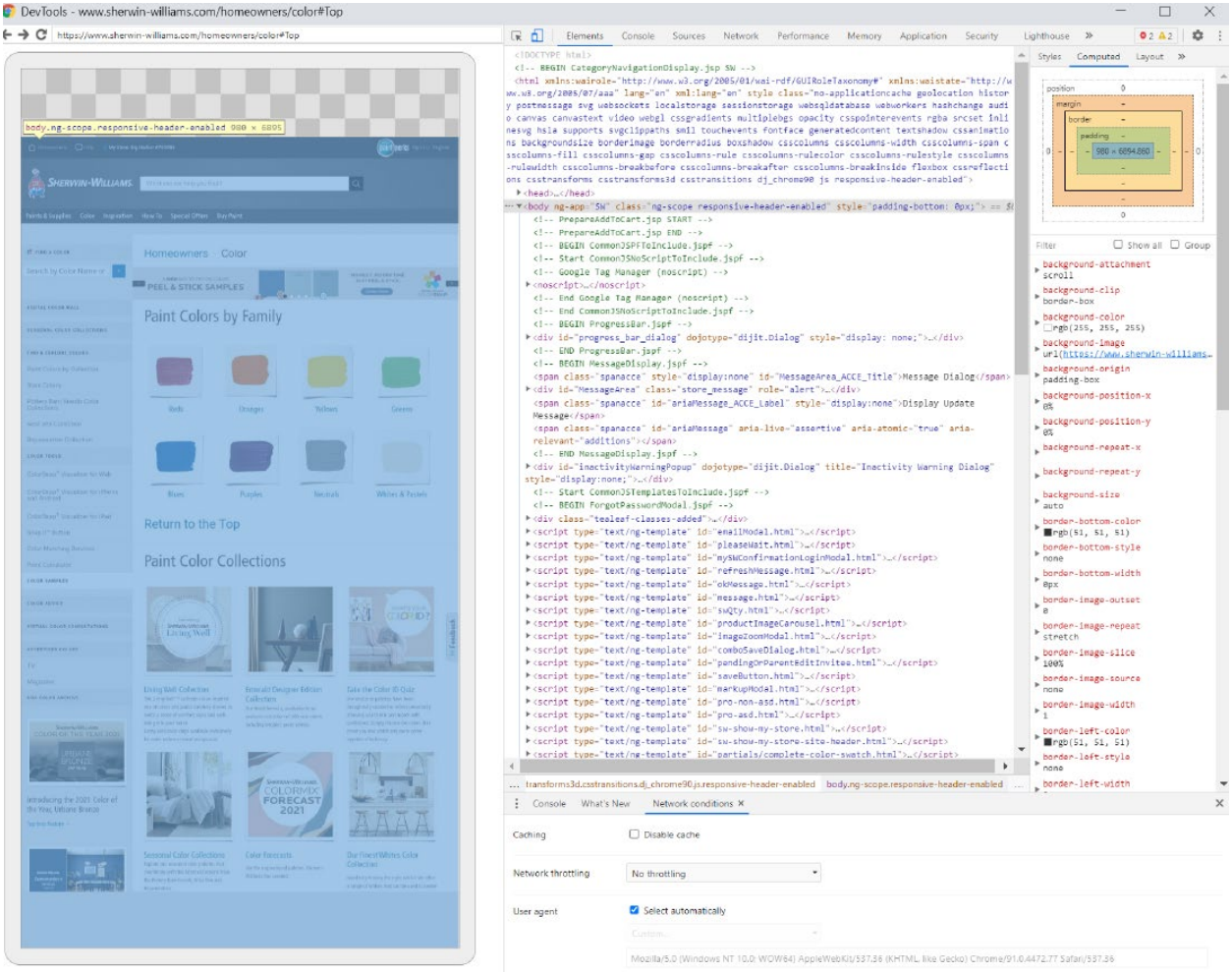
https://www.gsmarena.com/samsung_galaxy_s10-pictures-9536.php.

157. The Samsung Internet and Chrome browsers on the Accused Products enable users to request to access a webpage and, in response, retrieve HTML associated with the Web page having an original format defining an original page layout of the content on the Web page. The content retrieved includes text content, a plurality of hyperlinks, and a plurality of images. An example is shown below, where a user types a search query and is taken to the Sherwin Williams website in the Chrome browser.



158. Claim 1 of the ’889 patent recites “processing, via the wireless hand-held device, the markup language-based Web content with a rendering engine to generate page layout information corresponding to the original page layout of content on the Web page as interpreted by the rendering engine including a width of the Web page.” The Accused Products perform this operation.

159. The below image shows a screen capture of the Google Chrome Web Inspector connected to a Samsung S10 via a USB connection after requesting a Web page. The website is being displayed via Google Chrome after the markup language-based Web content has been processed with the Blink fork of the WebKit rendering engine. The bounding box for the entire page is shown as 980 x 6895 logical pixels (called Layout Units), for a width of 980 logical pixels.



160. Blink processes markup language-based Web content such as HTML to produce a document object model (DOM) tree and employs CSS rules and the CSS box model to generate a Layout Tree with page layout. Many of the core rendering processes employed by WebKit and Blink are similar as Blink and WebKit share the same core rendering code (WebCore). Blink uses a separate CSS layout engine called LayoutNG and uses a separate JavaScript engine. Aspects of

the main flow for Blink are similar to the main flow for WebKit below, including use of HTML and CSS parsers and generation of a Document Object Model (DOM) and DOM Tree.

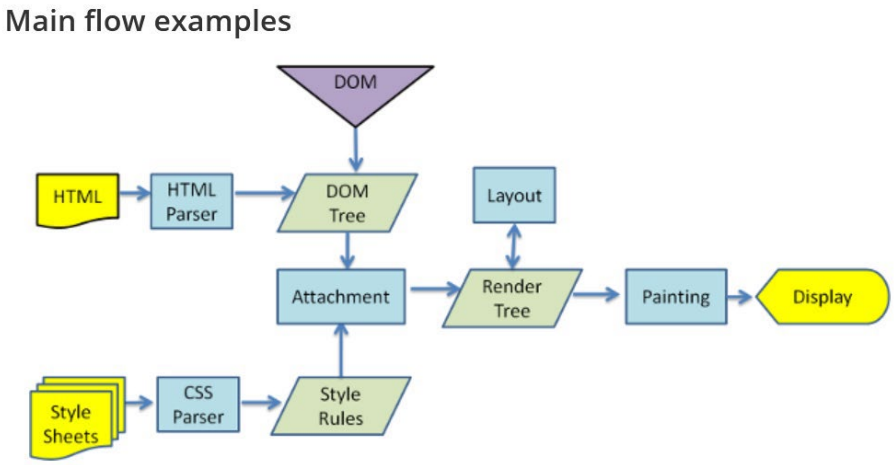


Figure : WebKit main flow

https://www.html5rocks.com/en/tutorials/internals/howbrowserswork/.

161. LayoutNG generates a Layout tree with bounding boxes (called boxes for short) employing the CSS box model.

CSS Box model

The CSS box model describes the rectangular boxes that are generated for elements in the document tree and laid out according to the visual formatting model.
Each box has a content area (e.g. text, an image, etc.) and optional surrounding padding, border, and margin areas.

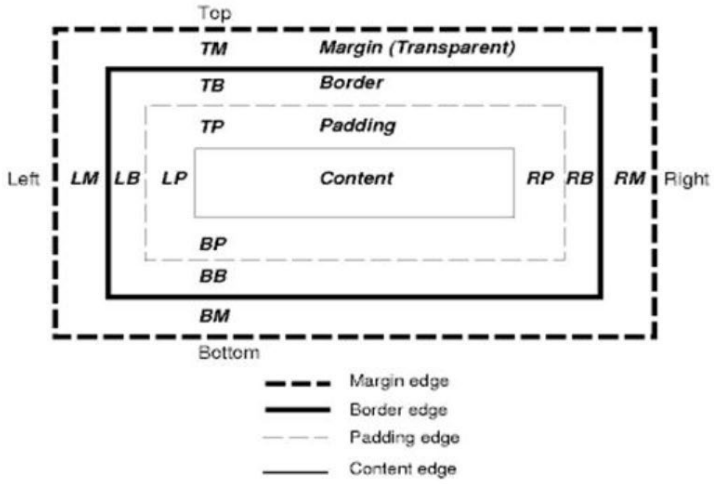


Figure : CSS2 box model

https://www.html5rocks.com/en/tutorials/internals/howbrowserswork/.

162. LayoutNG uses Logical & Physical Units.

Logical & Physical Units

To aid in readability and to ensure proper conversion between coordinate systems a set of dedicated units will be used to represent offsets, locations, sizes, and rects for each coordinate system where they are applicable. Conversion between coordinate systems must be done explicitly and no implicit conversion will be allowed.

Logical Units

NGLogicalOffset and NGLogicalSize represents an offset or a size. Each has a pair of LayoutUnits [11] that

represents the offset or size in the inline and block directions.

NGLogicalRect represents a rectangle and contains an offset and a size, represented by the two units described above.

These are all in a logical coordinate space in the sense that they do not take writing mode or directionality into account.

Physical Units

NGPhysicalOffset, NGPhysicalLocation, and NGPhysicalSize represents an offset, location or a size. The location unit is different from the offset one in that it represents the resolved location from the top of the tree while the offset unit represents the delta from the parent. Each has a pair of LayoutUnits that represents the offset or size in the horizontal and vertical directions.

NGPhysicalRect represents a rectangle and contains an offset and a size, represented by the two units described above.

These are all in a physical coordinate space in the sense that they represent coordinates in a top-to-bottom, left-to-right coordinate system. Converting from a logical unit to a physical one thus requires the coordinates to be resolved. The *Writing Modes and Coordinate Systems* section goes into more details.

https://docs.google.com/document/d/1uxbDh4uONFQOiGuiumlJBLGgO4KDWB8ZEkp7Rd47fw4/edit?tab=t.0#heading=h.guvbepjyp0oj.

163. LayoutNG generates a Layout Tree comprising NGFragment and NGText Nodes.

Layout Tree

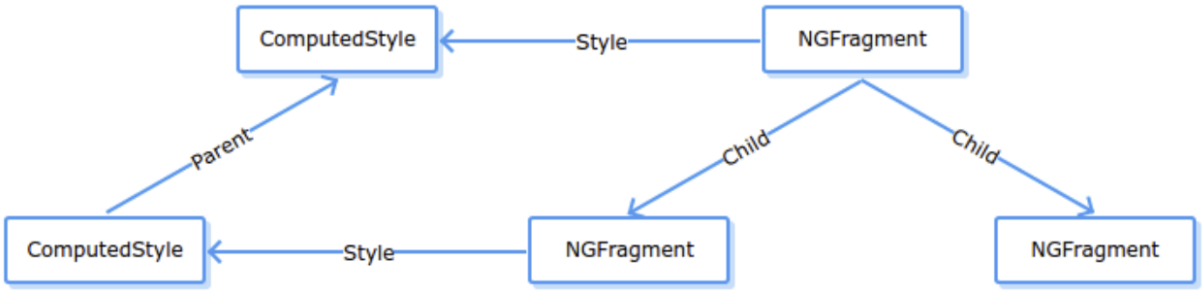


Figure shows the style tree (left) and layout tree (right) and their relationship.

The layout tree is made up entirely of NGFragment and NGText nodes and is immutable. Each node includes a back pointer to the computed style, a list of children and the size and location of the node itself. The location is relative to the immediate parent.

Given the immutability of the tree, use of relative coordinates, and the lack of a parent pointer entire subtrees may be reused across layouts. This concept is discussed in detail in the subsequent layouts section.

https://developer.chrome.com/docs/chromium/blinkng.

164. The Layout Tree also represents the bounding box hierarchy of the HTML document.

Pipelining style, layout, and pre-paint

Collectively, the rendering phases before *paint* are responsible for the following:

- Running the *style cascade* algorithm to calculate final style properties for DOM nodes.
- Generating the layout tree representing the box hierarchy of the document.
- Determining size and position information for all boxes.
- Rounding or snapping sub-pixel geometry to whole pixel boundaries for painting.
- Determining the properties of composited layers (affine transformation, filters, opacity, or anything else that can be GPU accelerated).
- Determining what content has changed since the previous paint phase, and needs to be painted or repainted (paint invalidation).

https://developer.chrome.com/docs/chromium/blinkng.

165. LayoutNG uses ClientRects to size and position HTML Elements (DOM objects) relative to the viewport.

ClientRect	Size and position of an Element relative to the viewport.

Walking up the Tree

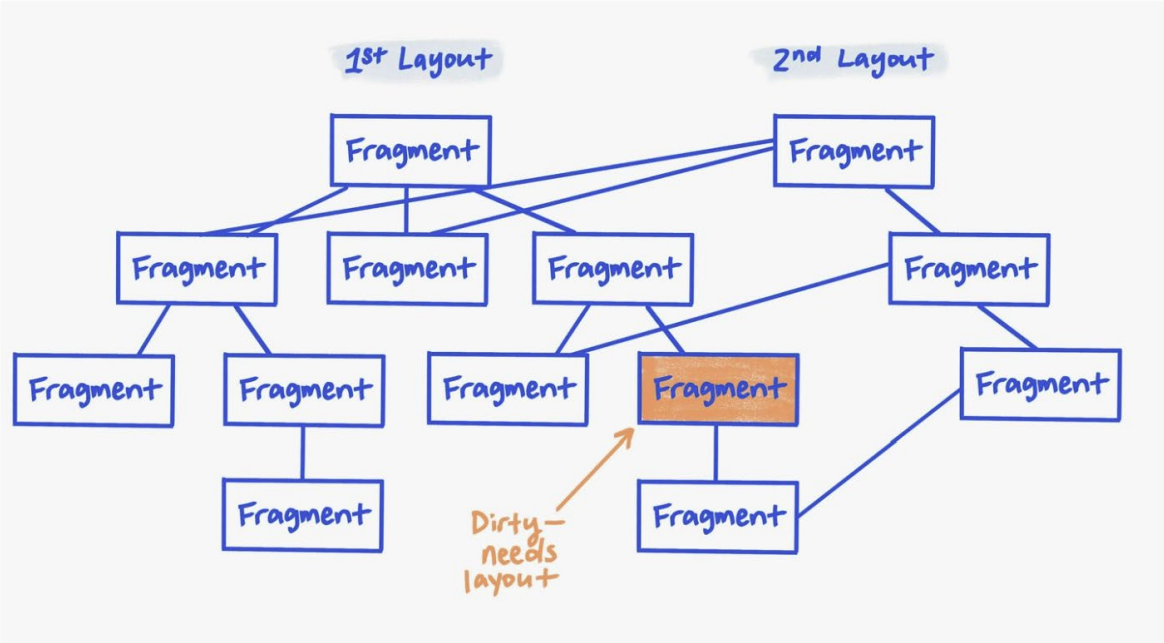
ClientRect calculations and IntersectionObserver need to perform a tree-walk up to the root in order to apply transform/layout information. Transform information is not on the layout tree. Adding layout-parent pointers in the Layout Tree would mean that we lose the immutability property; so instead the parent-layout pointers will be stored on the DOM tree. These operations will begin at the DOM node, ask its NGUnplacedFragment for layout information, then walk up the DOM tree to its parent-layout and so forth.

https://docs.google.com/document/d/1uxbDh4uONFQOiGuiumlJBLGgO4KDWB8ZEkp7Rd47fw4/edit?tab=t.0#heading=h.guvbepjyp0oj.

166. Blink generates an immutable fragment tree, which is used to represent the position and size of all elements on the page (without transforms applied).

The immutable fragment tree

The immutable fragment tree is the output of the layout stage of the rendering pipeline. It represents the position and size of all elements on the page (without transforms applied).

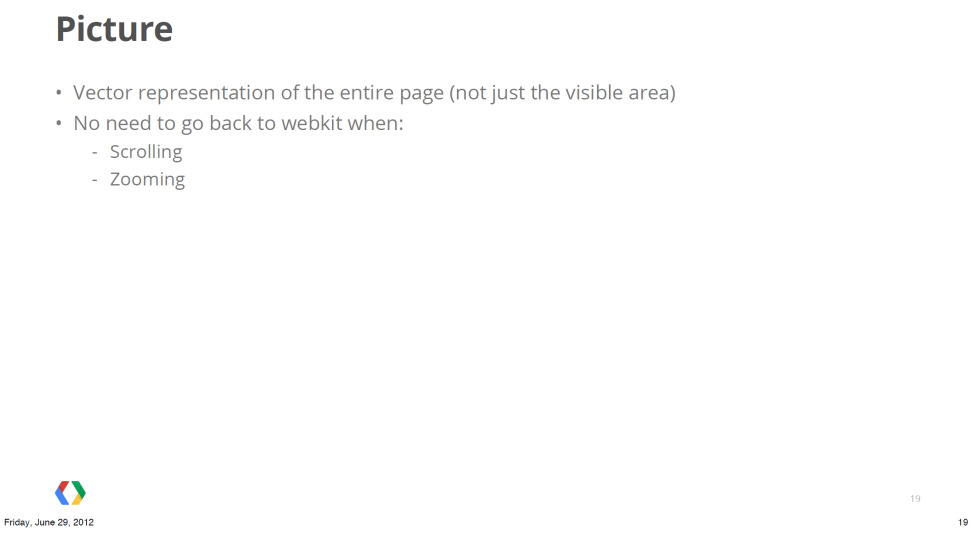


Each fragment represents a part of a DOM element. Typically there is only one fragment per element, but there can be more if it is split across different pages when printing, or columns when in a multi-column context.

https://developer.chrome.com/docs/chromium/renderingng-data-structures.

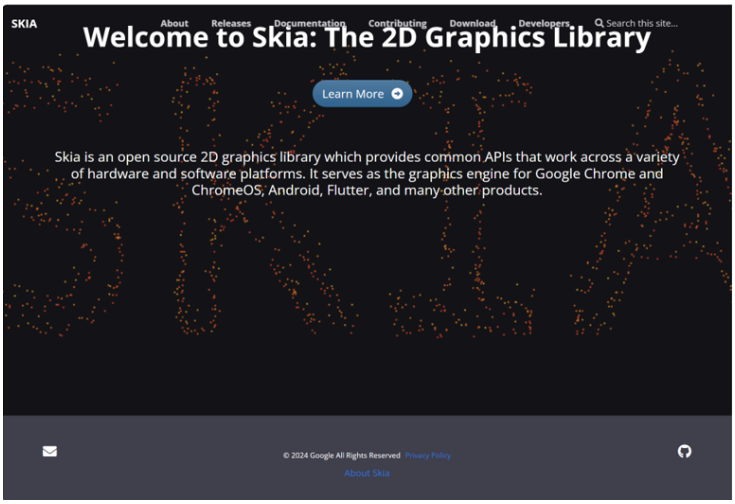
167. Claim 1 of the '889 patent recites "employing the page layout information to generate scalable page layout information" and "employing scalable page layout information and/or data derived therefrom to generate views of the Web page on the display, including, a view under which the width of the Web page is fully displayed; and views of the Web page generated in response to associated user inputs to enable a user to iteratively zoom in on a portion of the Web page while preserving the original page layout, functionality, and design of the content on the Web page defined by the markup language-based Web content." The Accused Products perform these operations.

168. WebView (and the Samsung Internet and Chrome browsers) creates a vector representation of the entire page, which is scalable.



Android WebView — Google IO 2012.

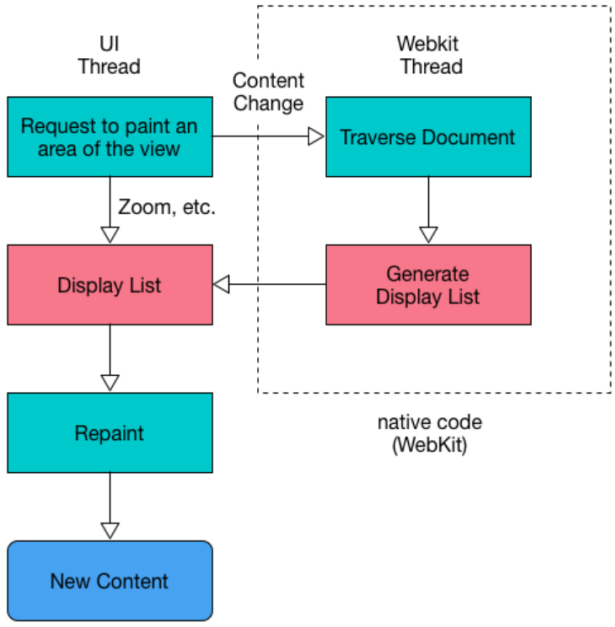
169. The Samsung Internet and Chrome browsers use the Skia 2D vector graphics rendering software/library, which is also referred to as the Skia Graphics Engine.



170. WebView (and the Samsung Internet and Chrome browsers) records lists of painting instructions for the full page and builds display lists. Zooming is accomplished by changing scale and translation parameters in a transformation matrix (SkMatrix) to scale and/or

translate position vector coordinates and calling the painting instructions in the display list. While the following describes the approach used by the Chrome browser with WebKit, the current painting process using Blink is similar. The display list model functions as a vector representation of the Web page, including for each object on the page such as text and images.

> So, instead of directly asking webkit to *paint* the screen, the team had a better idea. They decided to modify Webkit to *record* the list of painting instructions for the full page, but to not paint directly — basically using a "display list" model. The list of painting commands were for example "draw this text here", or "draw this image there".



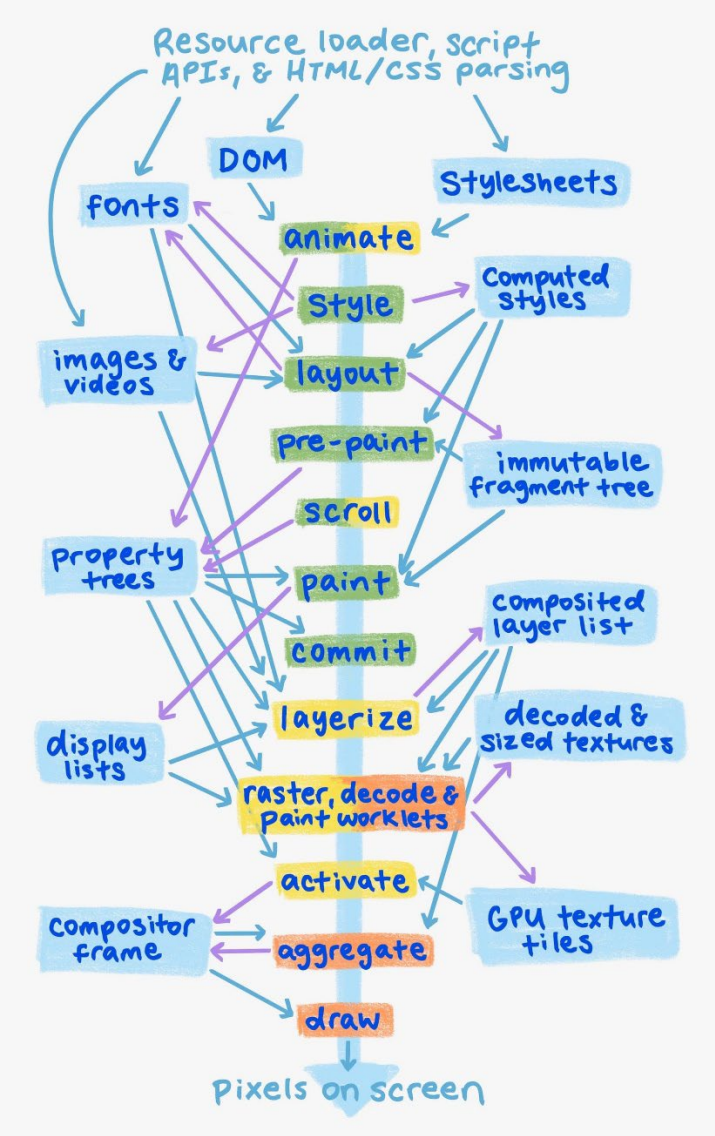
Rendering loop (Android)

> When the user would need to zoom in for example, it was not necessary to ask Webkit to repaint the screen — the webview instead could just replay the already present display list, only changing the zoom factor. This also gave a rather nice side-effect that the screen was always sharp — no text blurriness when zooming in, no missing content, as the content was always being redrawn "the right way".

> You could think of this display list as a way to have a vector representation of the entire page, that you could scale on demand and redraw — the difference between illustrator and photoshop.

https://medium.com/@camaelon/what-s-in-a-web-browser-83793b51df6c.

171. Display lists are used in the current Rendering Pipeline.



The stages are:

1. *Animate:* change computed styles and mutate *property trees* over time based on declarative timelines.
2. *Style:* apply CSS to the DOM, and create *computed styles*.
3. *Layout:* determine the size and position of DOM elements on the screen, and create the *immutable fragment tree*.
4. *Pre-paint:* compute property trees and invalidate any existing *display lists* and GPU *texture tiles* as appropriate.
5. *Scroll:* update the scroll offset of documents and scrollable DOM elements, by mutating property trees.
6. *Paint:* compute a display list that describes how to raster GPU texture tiles from the DOM.
7. *Commit:* copy property trees and the display list to the compositor thread.
8. *Layerize:* break up the display list into a *composited layer list* for independent rasterization and animation.
9. *Raster, decode and paint worklets:* turn display lists, encoded images, and paint worklet code, respectively, into GPU texture tiles.
10. *Activate:* create a *compositor frame* representing how to draw and position GPU tiles to the screen, together with any visual effects.
11. *Aggregate:* combine compositor frames from all the visible compositor frames into a single, global compositor frame.
12. *Draw:* execute the aggregated compositor frame on the GPU to create pixels on-screen.

https://developer.chrome.com/docs/chromium/renderingng-architecture#pipeline.

Display lists and paint chunks

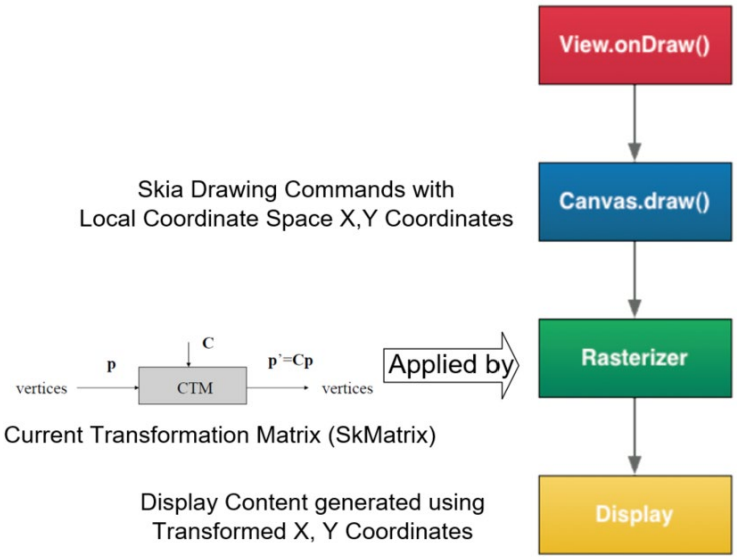
A display item contains low-level drawing commands (see here) that can be rasterized with Skia. Display items are typically simple, with just a few drawing commands, such as drawing a border or background. The paint tree walk iterates over the layout tree and associated fragments following CSS painting order to produce a display item list.

For example:



https://developer.chrome.com/docs/chromium/renderingng-data-structures#display_lists_and_paint_chunks.

172. Skia uses a Current Transformation Matrix (SkMatrix) for scaling and translation of content. An example is shown below for Skia painting.



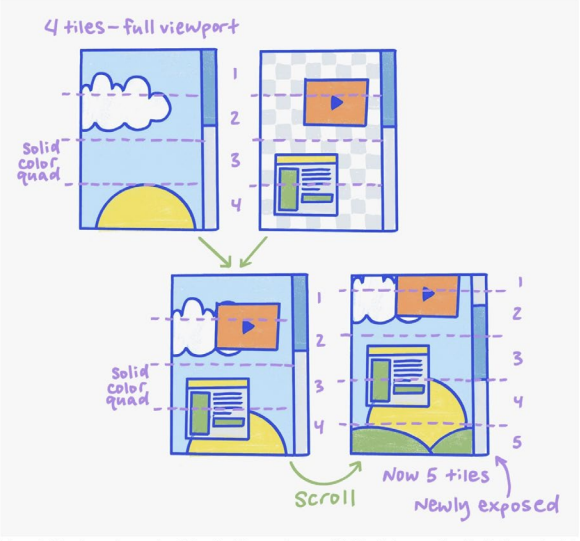
Android Accelerated Rendering Google IO 2011 (modified).

173. The rendering pipeline uses the Skia display lists to rasterize pixels onto GPU texture tiles.

Tiles

In theory, a render process or browser process compositor could rasterize pixels into a single texture the full size of the renderer viewport and submit that texture to Viz. To display it, the display compositor would just have to copy the pixels from that single texture into the appropriate position in the frame buffer (for example, the screen). However, if that compositor wanted to update even a single pixel it would need to re-rasterize the full viewport and submit a new texture to Viz.

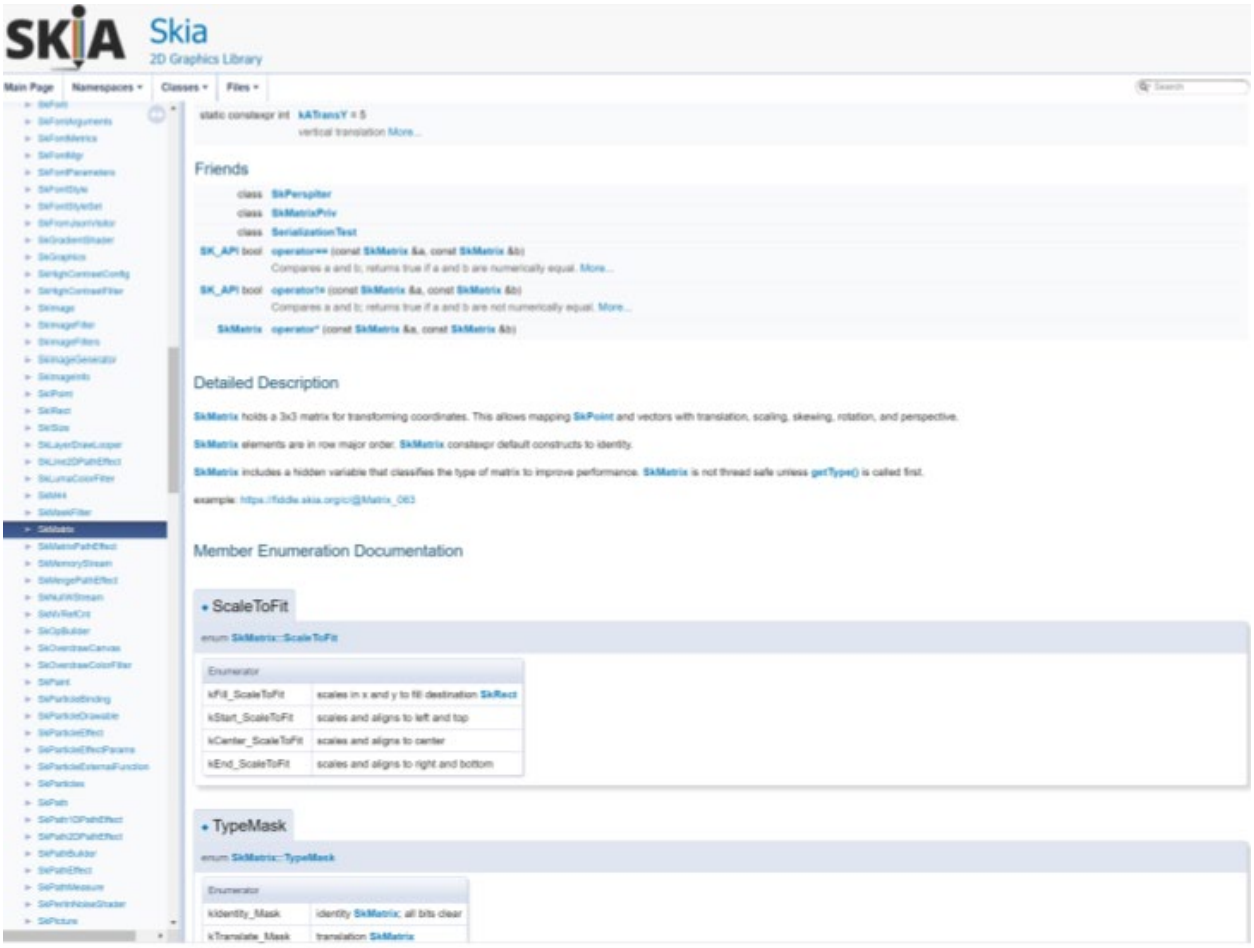
Instead, the viewport is divided into tiles. A separate GPU texture tile backs each tile with the rasterized pixels for part of the viewport. The renderer can then update individual tiles or even just change the position on screen for the existing tiles. For example, when scrolling a website, the position of existing tiles would shift up and only occasionally a new tile would need to be rasterized for content further down the page.



This image depicts an image of a sunny day, with four tiles. When a scroll occurs, a fifth tile begins to appear. One of the tiles happens to only have one color (sky blue), and there is a video and an iframe on top.

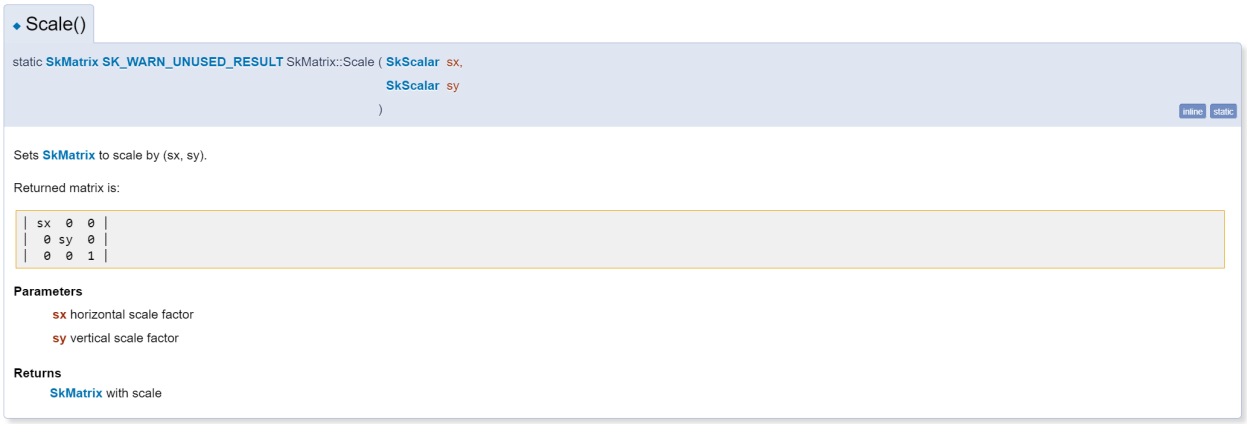
https://developer.chrome.com/docs/chromium/renderingng-data-structures#tiles.

174. The Display List uses the SKMatrix class to effect scaling. SKMatrix holds a 3x3 matrix for transforming coordinates (also known as an affine transformation). SKMatrix is used to effect translation and scaling. Skia renders images by ScaleToFit the destination rectangle (SKRect). SKRect is scaled using the current SKMatrix, and then the Image is scaled to fit the SKRect that is generated.



https://api.skia.org/classSkMatrix.html.

175. The Skia Scale() function can be used to change the sx (horizontal) and sy (vertical) scale factors in the SKMatrix. The setAffine() function can be used to set values for the first two rows of the SKMatrix. This enables scaling and translation to be performed together.



◆ Scale()

static SkMatrix SK_WARN_UNUSED_RESULT SkMatrix::Scale (SkScalar sx,
SkScalar sy
)

inline static

Sets SkMatrix to scale by (sx, sy).

Returned matrix is:

```
| sx  0  0 |
|  0 sy  0 |
|  0  0  1 |
```

Parameters

sx horizontal scale factor

sy vertical scale factor

Returns

SkMatrix with scale



◆ setAffine()

SkMatrix& SkMatrix::setAffine (const SkScalar affine[6])

Sets SkMatrix to affine values, passed in column major order.

Given affine, column, then row, as:

```
| scale-x  skew-x translate-x |
|  skew-y scale-y translate-y |
```

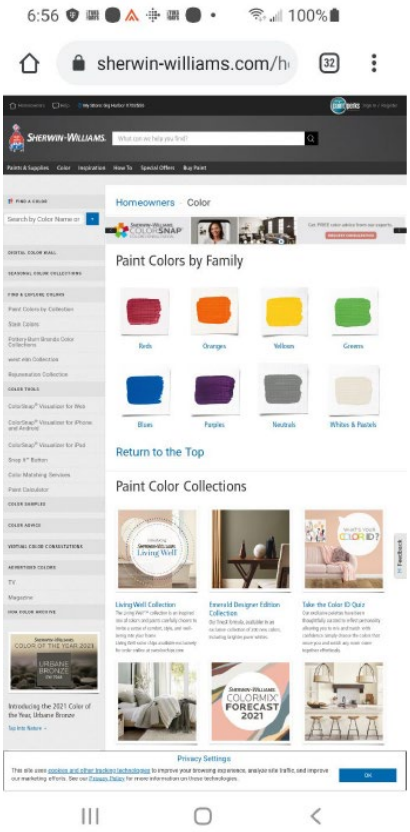
SkMatrix is set, row, then column, to:

```
| scale-x  skew-x translate-x |
|  skew-y scale-y translate-y |
|       0       0           1 |
```

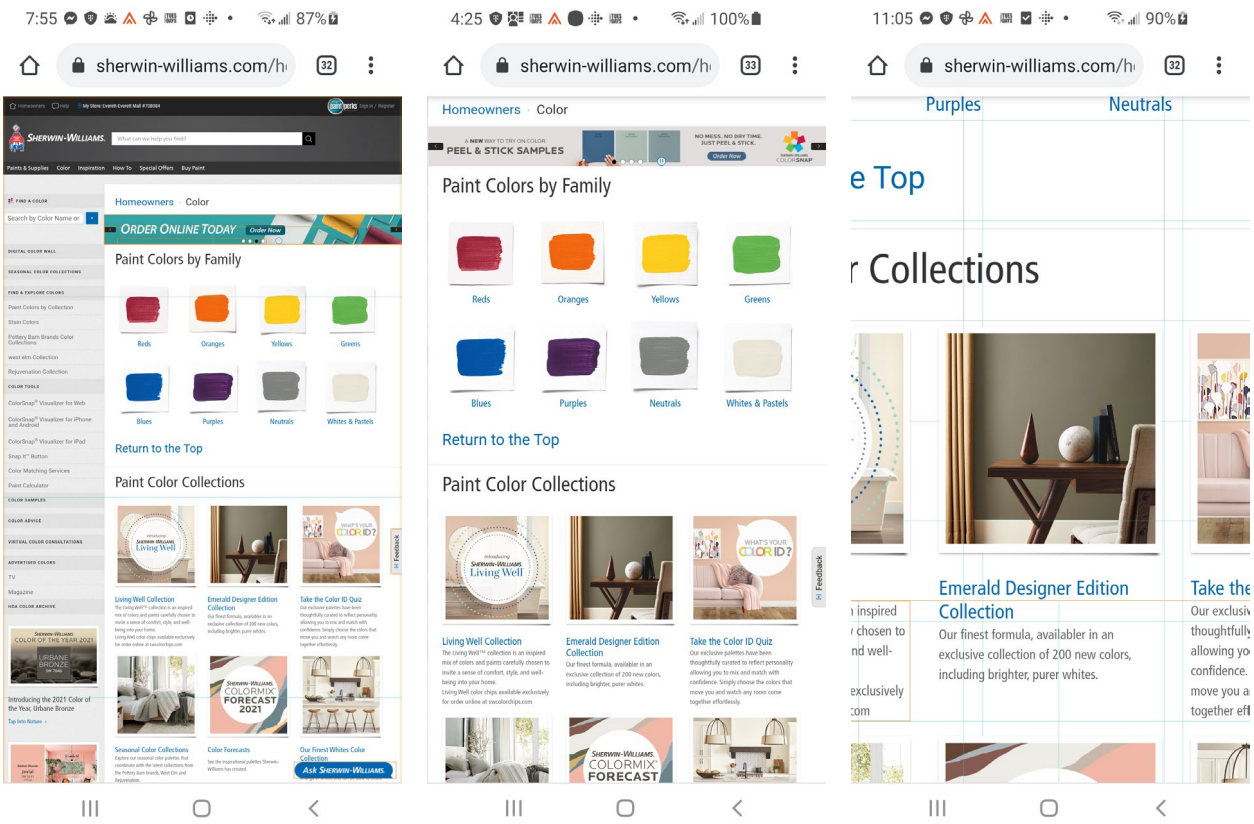
Parameters

affine 3 by 2 affine matrix

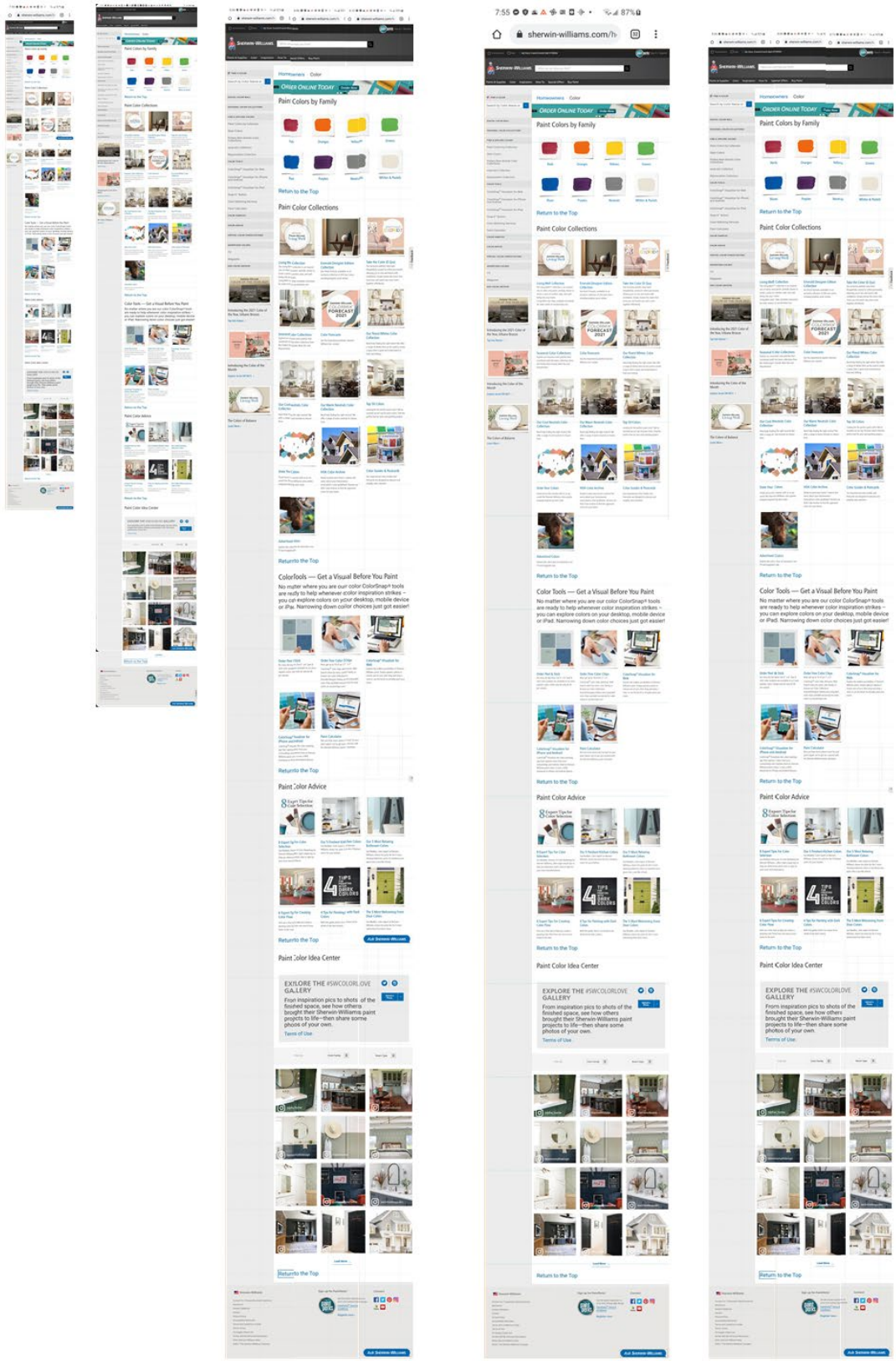
176. The screenshot below shows the Chrome browser on a Samsung Galaxy S10, which displays the Sherwin Williams website at full width when the page is initially loaded.

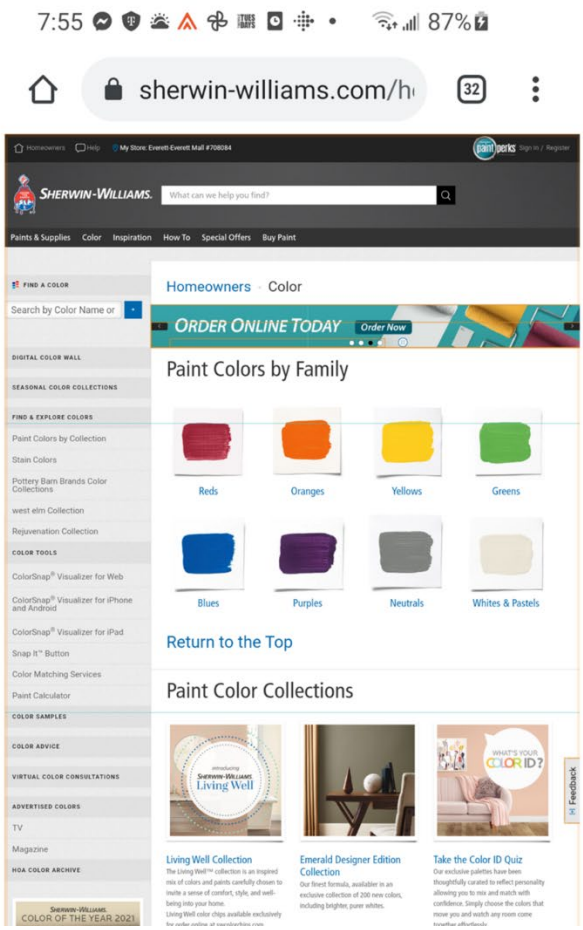


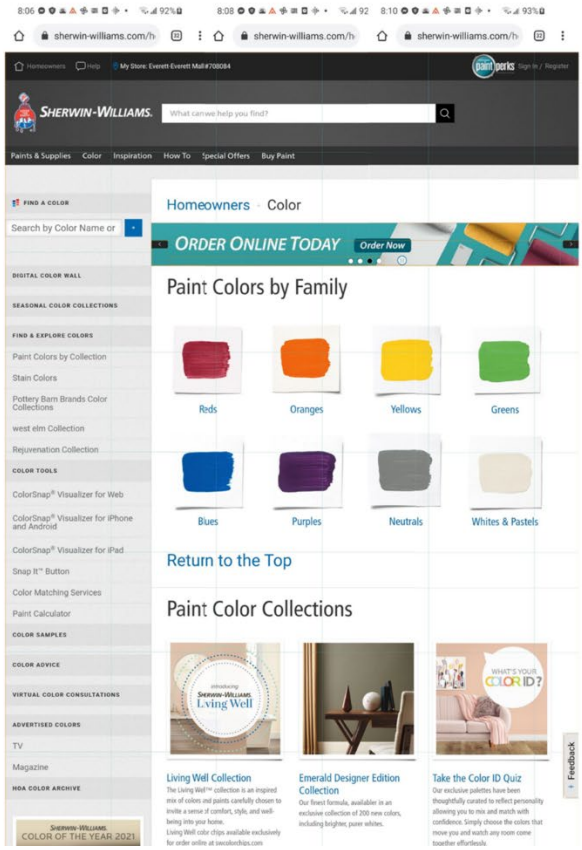
177. The screenshots below show the Web page as displayed initially (left), as displayed after the user has zoomed in on a portion of the Web page at a first zoom level (middle), and as displayed after the user has further zoomed in on a portion of the Web page (right). The left and right screenshots further show outlines for the tiles in cyan. The user is able to iteratively zoom in on a portion of the Web page while preserving the original page layout, functionality, and design of the content on the Web page defined by the markup language-based Web content through use of vector graphics combined with scalable page layout information generated using the CSS box model.



178. The three screenshot mosaics at left below demonstrate preservation of the original page layout, functionality, and design of the content on the Web page defined by the markup language-based Web content at three zoom levels, while the pair of screenshot mosaics at right below show a normalized comparison between the first and third zoom levels demonstrating perfect scaling. The third image shows the top portion of the normalized mosaics.

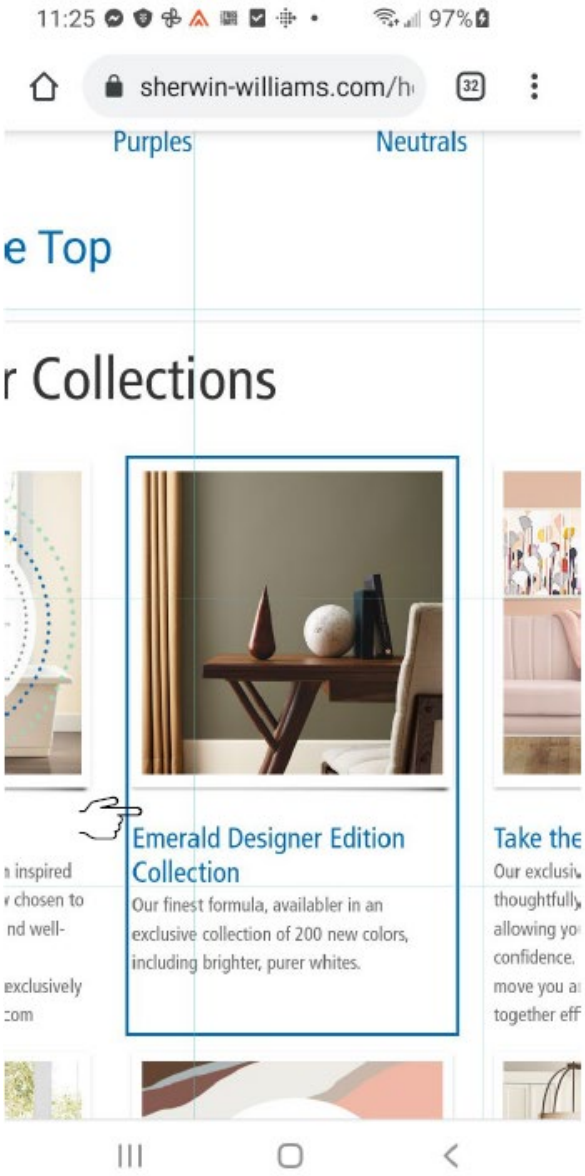


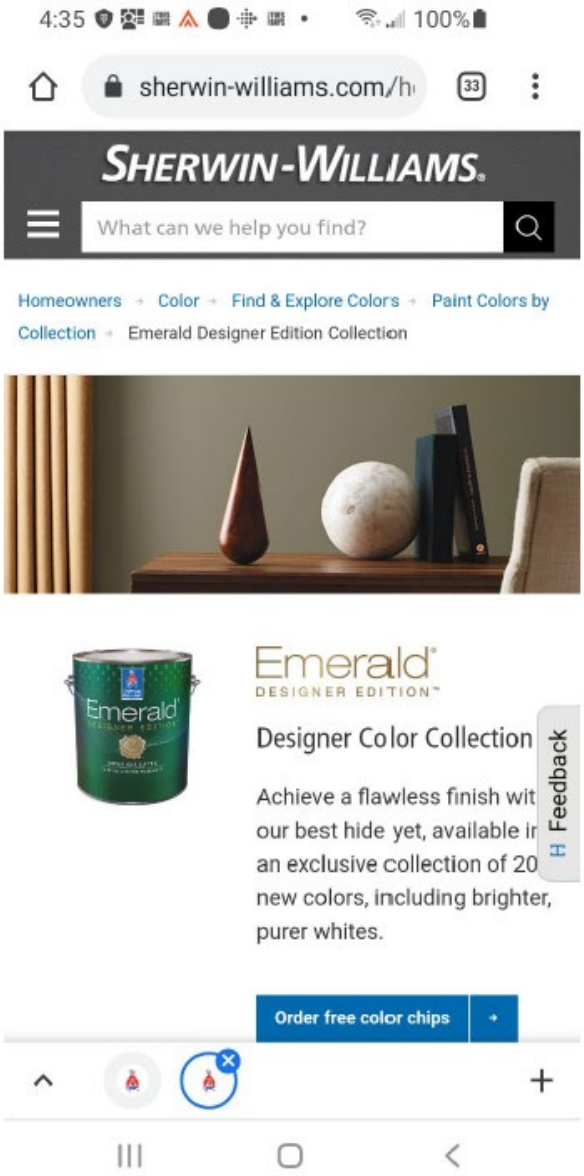




179. Claim 1 of the ’889 patent recites “wherein preservation of the functionality defined by the markup language-based content includes preservation of hyperlink functionality.” The Accused Products satisfy this requirement.

180. The images below show hyperlink functionality is preserved when zoomed in. In the first image, a hyperlink is clicked while the page is zoomed in. In the second image, the Web page corresponding to the hyperlink is displayed.





181. Claim 5 further recites:

The method of claim 1, further comprising:
parsing and processing markup language code with the rendering engine to determine the original page layout of display content within the Web page as interpreted by the rendering engine, wherein the original page layout defines a layout location for a plurality of objects, including text objects associated with the text content and image objects associated with the plurality of images included in the Web page; and
for each object,

defining an object datum corresponding to the layout location for the object on the original page layout; and
associating the object and its object datum.

182. The Accused Products perform these operations.

183. The Blink rendering engine parses and processes markup language code such as HTML and produces a DOM and a Layout Tree that define a layout location for a plurality of objects, including text objects and image objects. Each HTML element corresponds to an object in the DOM that has an associated ClientRect, which is a rectangle with a location and size.

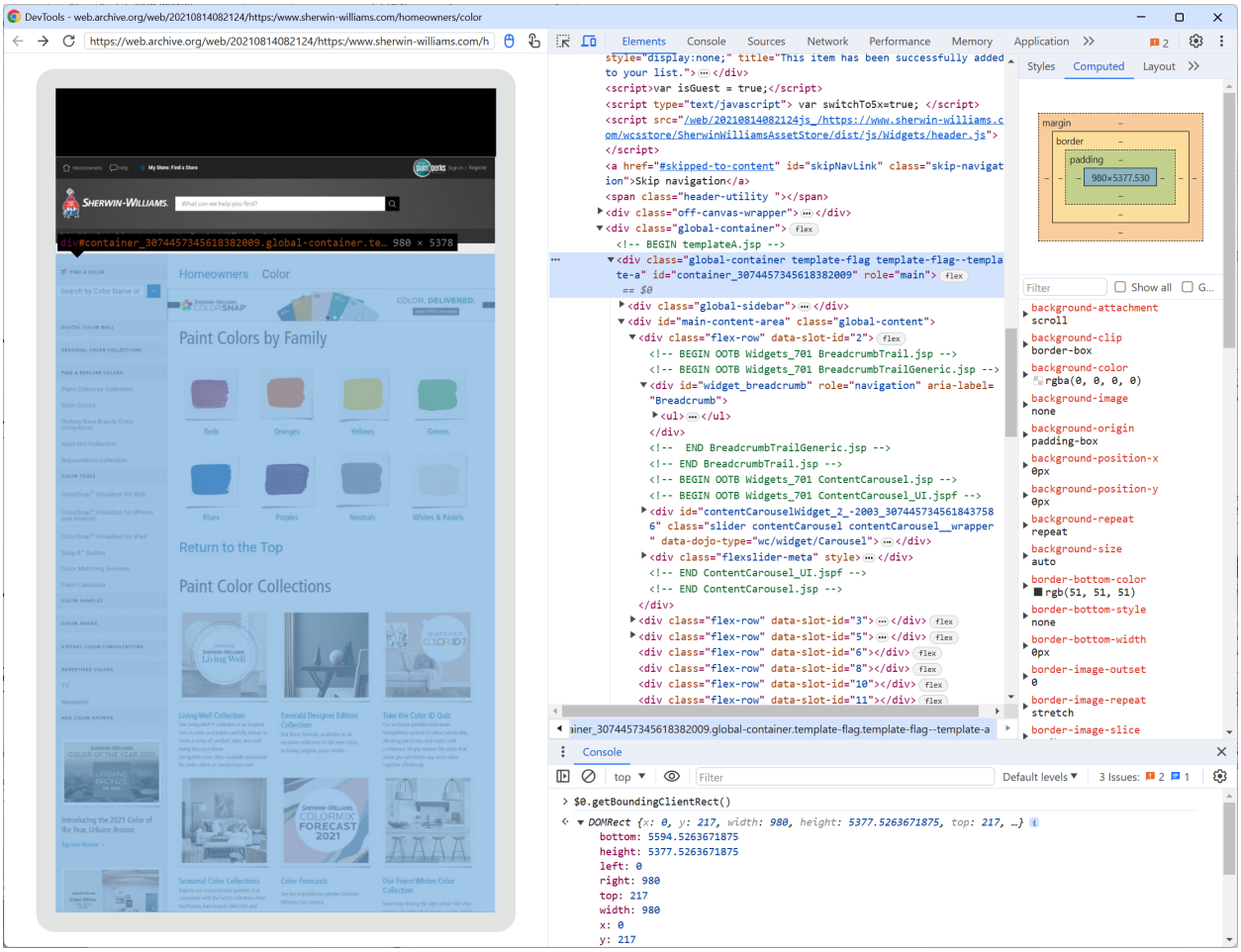
ClientRect	Size and position of an Element relative to the viewport.

https://docs.google.com/document/d/1uxbDh4uONFQOiGuiumlJBLGgO4KDWB8ZEkp7Rd47fw4/edit?tab=t.0#heading=h.guvbepjyp0oj.

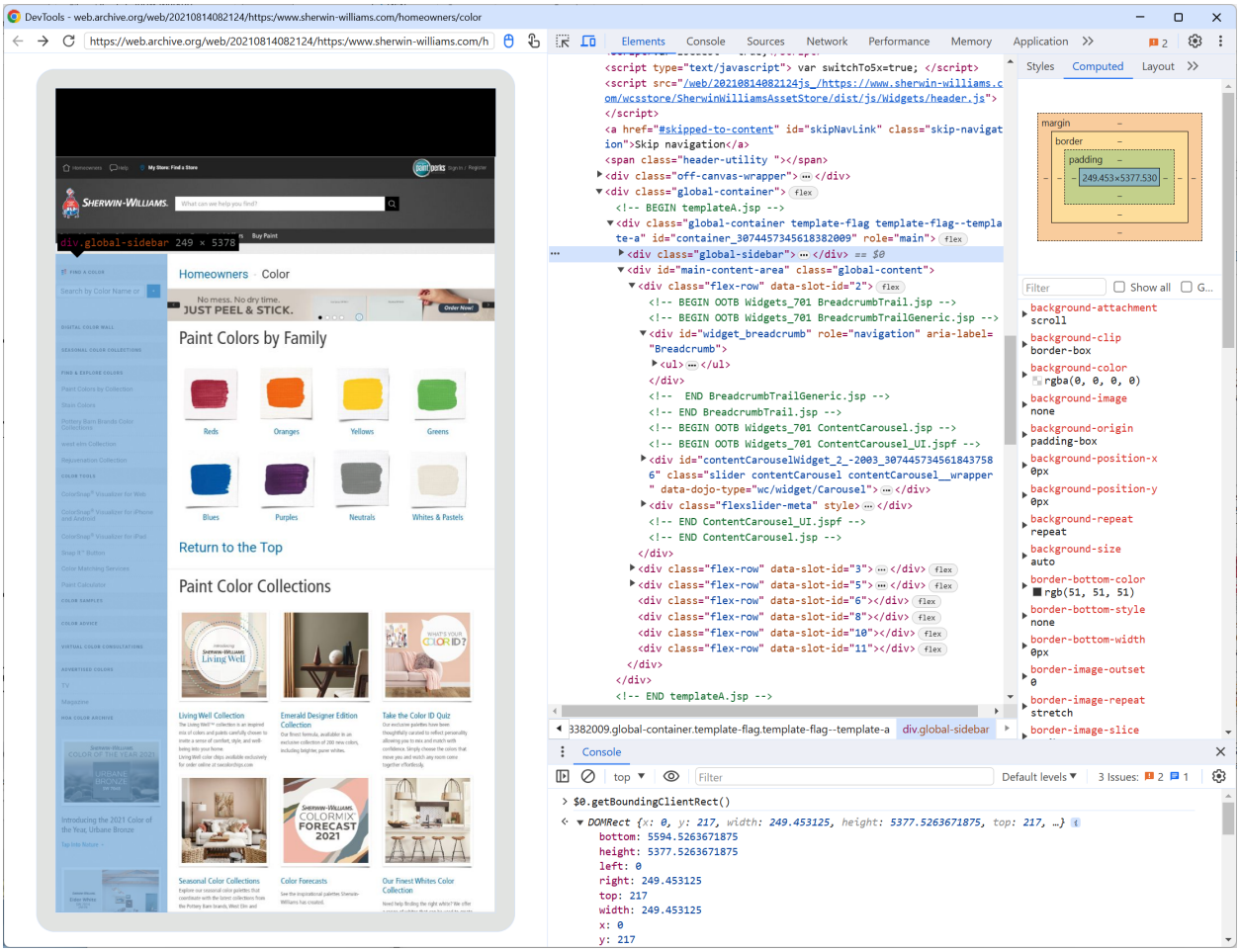
184. Each ClientRect has a defined object datum corresponding to the layout location for the object relative to the viewport origin (upper left-hand corner) on the original page layout. The object datums comprise x, y coordinates in the local coordinate space, which is a resolution-independent vector space; each x, y coordinate in the local coordinate space is a position vector, with the tail of each position vector at 0,0 origin (upper left-hand corner) and the head of the position vector at the x, y coordinate. The object datums (x, y coordinates) are used to support a vector representation of the Web page that can be scaled and translated to enable the user to zoom in and out views of the Web page with perfect scaling.

185. The following examples employ a Samsung Galaxy S10 connected via USB to a host laptop computer running the Google Chrome browser and the Chrome Web Inspector. An archive.org version of the Web page is used to represent what the Web page looked like on August 14, 2021. Each of the screenshots show the bounding box for the ClientRect corresponding to the selected (highlighted in light blue) element from the DOM. This bounding box is also referred to as a DOMRect. This first example shows the bounding box for the ClientRect for the "global

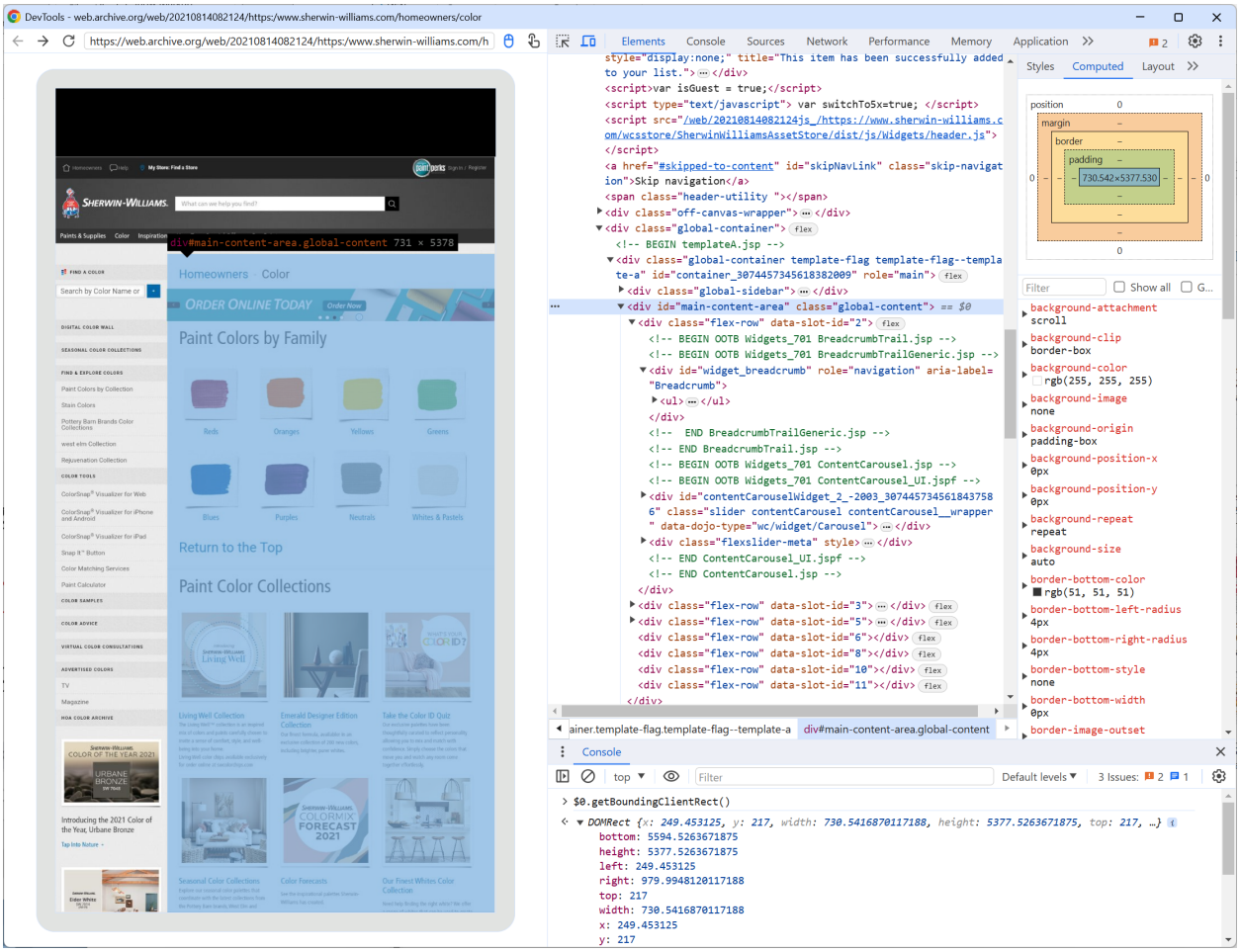
container" div class. The Layout Units have a resolution of 1/64 pixels, which is reflected in the non-integer bottom and height.



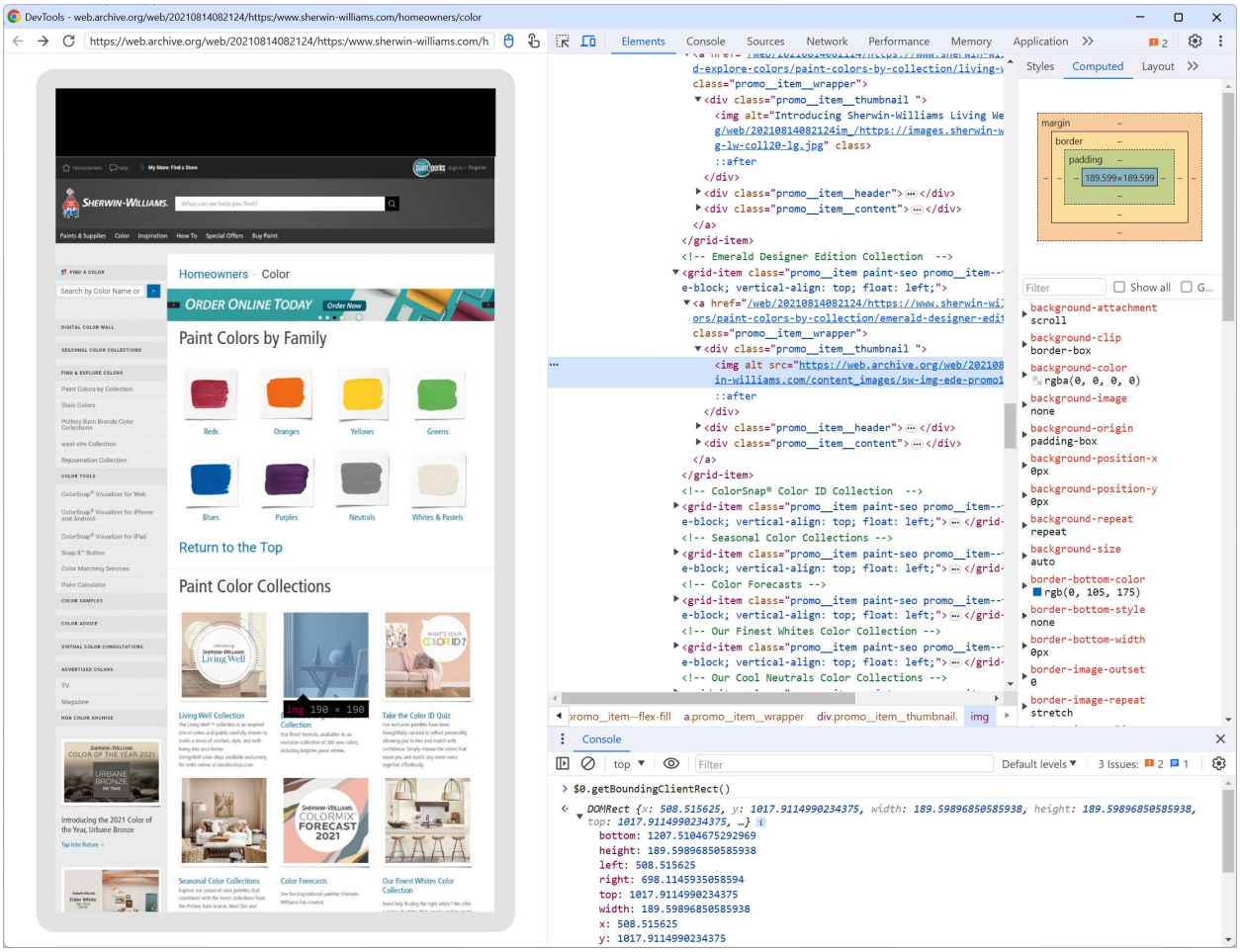
186. This example shows the ClientRect bounding box and DOMRect for the "global sidebar" div element. The DOMRect has an object datum of (0,217).



187. This example shows the ClientRect bounding box and DOMRect for the “main-content-area” div element.



188. This example shows the ClientRect bounding box and DOMRect for the highlighted image element.

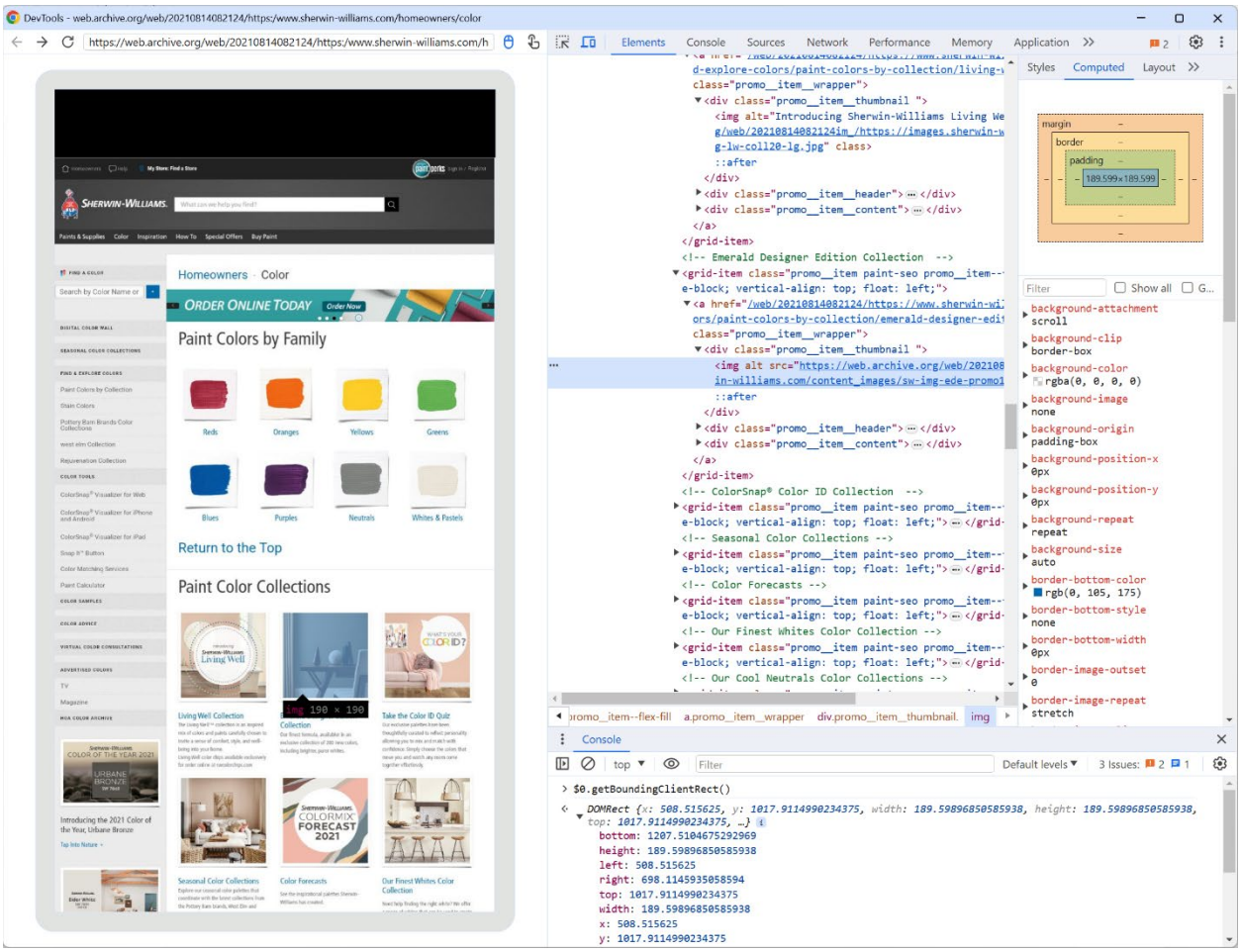


189. Each ClientRect has a defined object datum corresponding to the layout location for the object relative to the viewport origin (0,0). The Blink Rendering engine stores the ClientRect bounding box and DOMRect for each element in the DOM that has associated display content. Each object has an associated x, y coordinate comprising the object datum for the object.

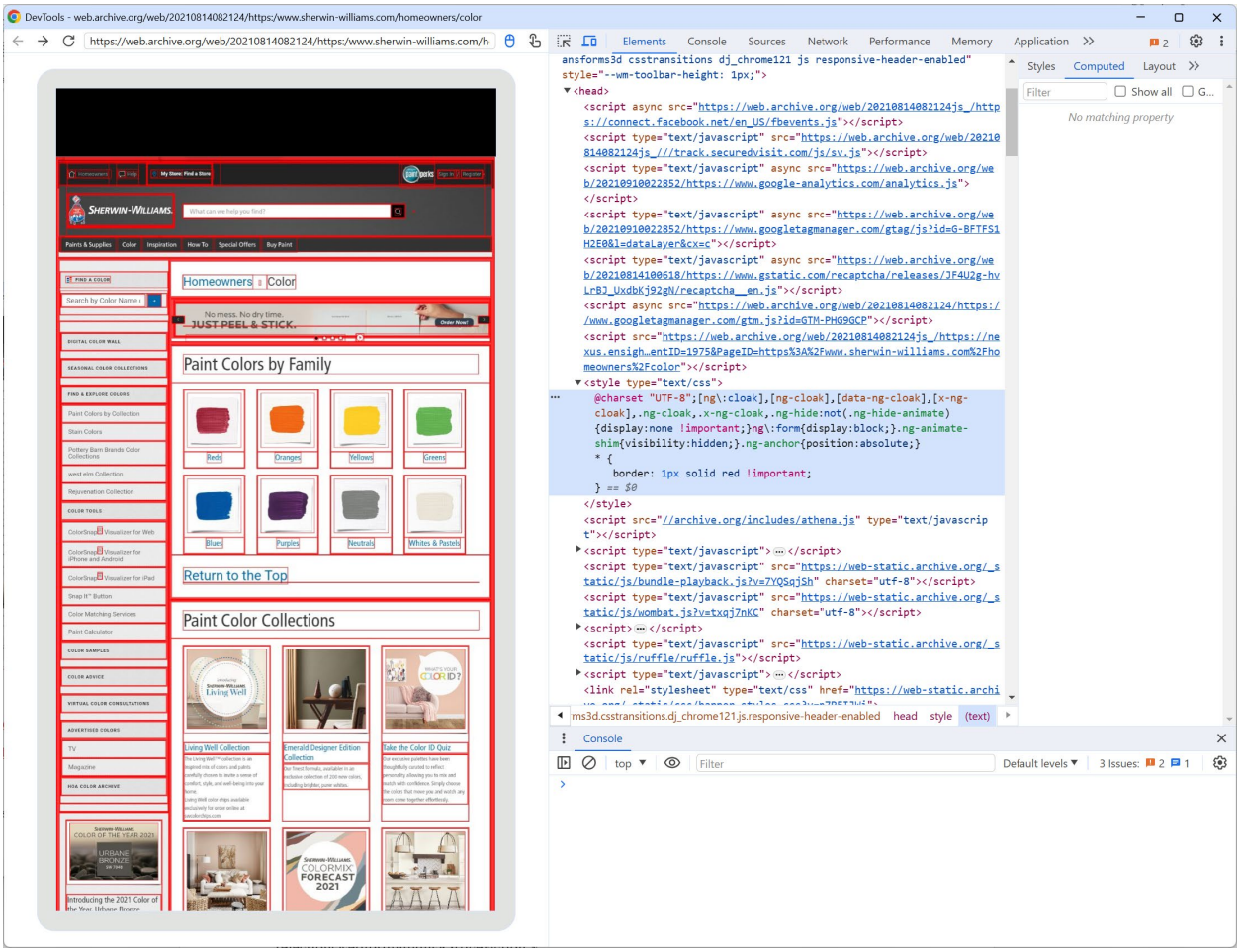
ClientRect	Size and position of an Element relative to the viewport.

https://docs.google.com/document/d/1uxbDh4uONFQOiGuiumlJBLGgO4KDWB8ZEkp7Rd47fw4/edit?tab=t.0#heading=h.guvbepjyp0oj.

190. The object datum for the image highlighted in the screenshot below is 508.515 (x), 1017.911 (y) in LayoutUnits.



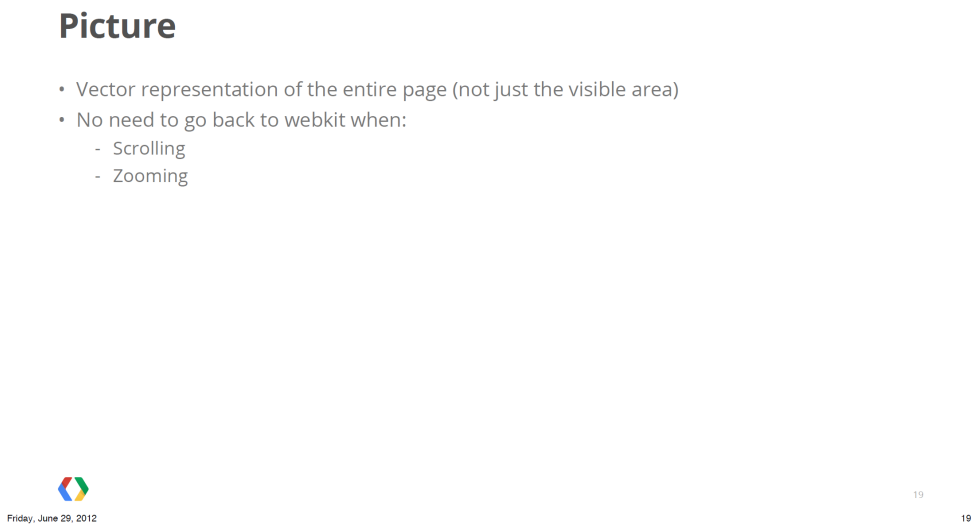
191. A visual representation of the Layout Tree structure can be shown by modifying the HTML document to render a visual border for each bounding box.



The following code is added to the end of the <style> section highlighted above:

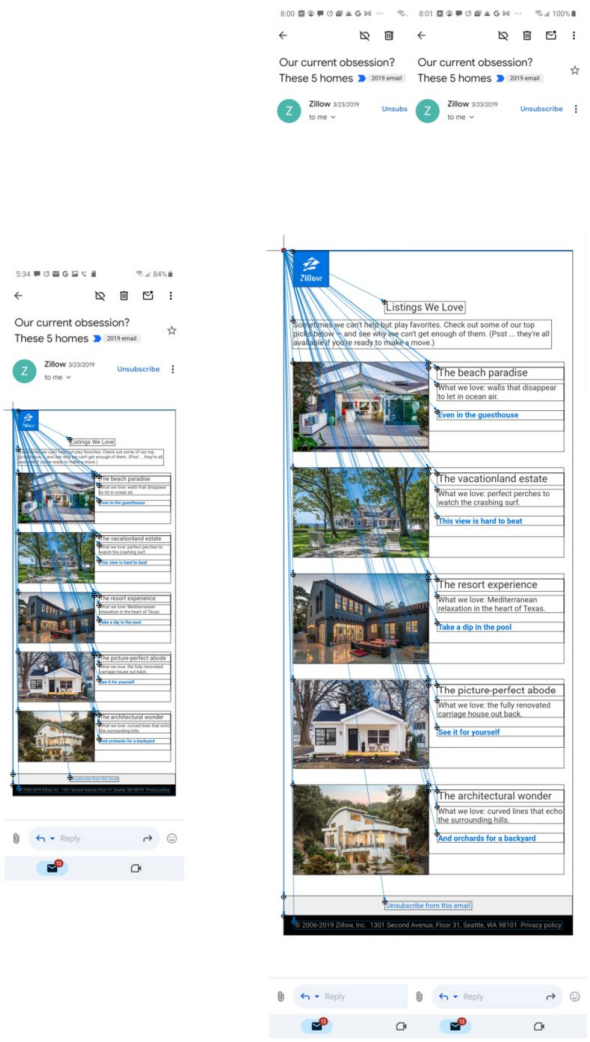
```
* {
  border: 1px solid red !important;
}
```

192. WebView (and the Samsung Internet and Chrome browsers) creates a vector representation of the entire page, which is scalable.



Android WebView — Google IO 2012.

193. The following illustrates vector-based scaling of the page layout for a Web page using the Android Chrome browser. This example shows vectors for selected content. The tail of each vector is at the page and local coordinate space origin (0,0) and the head of each vector is at the datum for an associated object.



Samsung's Inducement of Infringement of the '889 Patent

194. SoftView incorporates by reference and re-alleges the foregoing paragraphs of this Complaint as if fully set forth herein.

195. Defendants were served on October 5, 2011, with the Second Amended Complaint in the Delaware litigation alleging infringement of the '353 and '926 patents. The '889 patent issued on November 30, 2010, before service of the Second Amended Complaint, and is in the same family as the '353 and '926 patents.

196. SoftView is informed and believes, and thereon alleges, that Defendants have actively induced infringement of at least claim 5 of the '889 patent in violation of 35 U.S.C. §

271(b) by knowingly and intentionally encouraging or aiding third parties to use infringing software and hardware products in this judicial district and elsewhere throughout the United States, without license or authority from SoftView, including at least the Accused Products. SoftView is informed and believes, and thereon alleges, that these third parties have infringed at least claim 5 of the '889 patent in violation of 35 U.S.C. § 271(a) by using infringing software and hardware products, including the Accused Products. The Defendants through at least their user manuals, Web site support and training materials actively induced their customers and users of the Accused Products to infringe the '889 patent. Furthermore, the natural and intended use of the Accused Products (*e.g.*, browsing the internet with Samsung Internet and Chrome) infringes at least claim 5 of the '889 patent. Defendants know and intend that the Accused Products are used in this infringing manner. In addition, SoftView reasserts evidence presented in Count I *supra* supporting that Defendants knowingly induce infringement of at least claim 5 of the '889 patent.

197. Defendants have induced infringement of at least claim 5 of the '889 patent.

198. On information and belief, Defendants have committed acts of infringement notwithstanding actual knowledge of the '889 patent and without a good faith basis to believe that its activities do not infringe at least claim 5 of the '889 patent. All infringement of the '889 patent following Defendants' knowledge of the '889 patent is willful, and SoftView is entitled to treble damages and attorneys' fees and costs incurred in this action under 35 U.S.C. §§ 284 and 285.

199. In the event Defendants assert they were not aware of the'889 patent or lack the requisite knowledge and intent to induce infringement, Defendants would have been willfully blind of SoftView's patent rights.

200. In view of the Delaware litigation and the '889 patent that issued before the litigation from the same family as the '353 and '926 patents, Defendants were aware of a high

probability of the existence of the ’889 patent and their infringement of it, and/or would had to have taken deliberate actions to avoid learning of the existence of the ’889 patent and their infringement of it. In addition, the related ’729 patent was cited to Defendant SEA in patent applications, which further shows Defendants either knew about the existence and infringement of other patents such as the ’889 patent or were willfully blind to their existence and infringement of them.

201. Defendants’ actions are at least objectively reckless as to the risk of infringement and this objective risk was either known or should have been known by Defendants.

202. Defendants’ direct and indirect infringement of the Asserted Patents is and has been willful, intentional, deliberate, and/or in conscious disregard of SoftView’s rights under the Asserted Patents.

203. SoftView has incurred substantial damages, including monetary damages.

204. SoftView has been irreparably harmed by Defendants’ infringement of the ’889 patent.

205. Therefore, SoftView is entitled to actual and/or compensatory damages, reasonable royalties, pre-judgment and post-judgment interest, enhanced damages, and costs.

COUNT III – INFRINGEMENT OF U.S. PATENT NO. 10,083,154

206. SoftView incorporates by reference and re-alleges the foregoing paragraphs of this Complaint as if fully set forth herein.

207. Defendants have directly infringed at least claim 13 of the ’154 patent.

208. As a matter of law, the notice provisions of 35 U.S.C. § 287 do not apply in this case for the ’154 patent, and therefore SoftView has satisfied its obligations under 35 U.S.C. § 287 with respect to the ’154 patent.

209. At the time of the claimed invention, reproducing web pages on a variety of devices presented a significant technical problem. Most internet content displayed as a flat single resolution with no browser support for zoom. '353 Patent, 2:1-3. Much of the content on the internet was designed for display on desktop computers with a single target resolution (e.g., 640x480, 800x600). '353 Patent, 2:4-5. Major internet content providers chose to create their Web pages using fixed resolution structures, such as tables, which gave them the ability to control the look and feel of their Web sites. '353 Patent, 2:5-10. The fixed resolution approach was so ubiquitous that it became the most common method to brand or uniquely identify websites. '353 Patent, 2:10-13.

210. While the fixed resolution approach was useful for site branding and product differentiation, it presented a daunting technical problem: how to display Internet content designed for desktop computers on devices with smaller screens, lower resolution, and/or different aspect ratios, such as cell phones and handheld computers. '353 Patent, 2:13-18. For example, a website designed for the fixed resolution of a larger desktop display would not fit correctly on small screens, low resolution screens, or devices with different aspect ratios.

211. The '154 patent claimed invention provides a solution to this daunting technical problem. Claim 13 recites as follows:

> 13. A method performed by a mobile hand-held device including a touch-sensitive display and an HTML rendering engine, comprising:
> receiving an HTML document including HTML code and cascading style sheet (CSS) code and content associated with the HTML document, the HTML code comprising a plurality of HTML elements including at least one HTML paragraph element, at least one HTML image element, and at least one HTML hyperlink element;
> processing the HTML document with the HTML rendering engine to render a first representation of the HTML document having an interpreted page layout, functionality, and design of the content associated with the HTML document that is in accordance with the HTML code and CSS code, wherein rendering the first representation of the HTML document includes,

parsing the HTML document to identify the plurality of HTML elements;
logically grouping content associated with HTML elements into HTML objects;
generating page layout information including a bounding box for each HTML object; and
storing information that links each HTML object with its corresponding page layout information,
wherein the page layout information further includes information from which a page layout location of each of the bounding boxes can be determined;
translating the first representation of the HTML document to generate a scalable vector representation of the HTML document wherein generating the scalable vector representation includes,
defining a primary datum corresponding to the interpreted page layout;
for each HTML object,
defining an object datum corresponding to a layout location datum for the HTML object's bounding box;
generating a vector from the primary datum to the object datum for the HTML object's bounding box; and
creating a reference that links the HTML object to the vector that is generated; and
render the scalable vector representation of the HTML document on the touch-sensitive display using a first scale factor to display the HTML document at a first zoom level under which the HTML document is displayed to fit across a width of the touch-sensitive display,
wherein the scalable vector representation of the HTML document is configured to enable a user to view the HTML document at one or more user-defined zoom levels by rendering the scalable vector representation on the touch-sensitive display using one or more respective scale factors in response to associated user inputs made via the touch-sensitive display, and wherein the interpreted page layout, functionality, and design of the content associated with the HTML document is preserved at each of the first zoom level and the one or more user-defined zoom levels.

212. As claimed in claim 13 of the ’154 patent, an HTML document including HTML code and CSS code is processed to generate a scalable vector representation of the Web page. At the time of the claimed invention, this process was not well-known, routine, or conventional. Nor was it well-known, routine, or conventional to render a web page at multiple scale factors.

213. The scale factor and scalable vector representation of the claimed invention permits websites to be displayed so that they properly fit the width of smaller screens or those with different

aspect ratios and resolutions, such as smartphones, while preserving the interpreted page layout, functionality, and design of the content associated with the HTML document at multiple user-defined zoom levels.

214. Claim 13 provides additional details related to the generation of the scalable vector representation, including defining a primary datum corresponding to the interpreted page layout; for each HTML object, defining an object datum corresponding to a layout location datum for the HTML object's bounding box; generating a vector from the primary datum to the object datum for the HTML object's bounding box; and creating a reference that links the HTML object to the vector that is generated. At the time of the claimed invention, it was not well-known, routine, or conventional to do this. The claim recites an object datum corresponding to a layout location datum for the HTML object's bounding box, and a primary datum, that permits vectors to be defined between the object and primary datum, thereby allowing a vectorized view of the Web page. An example is shown in the exemplary Figure 4C of the patent specification, where the primary datum is in the upper left corner of the screen and each arrow points to an object datum.

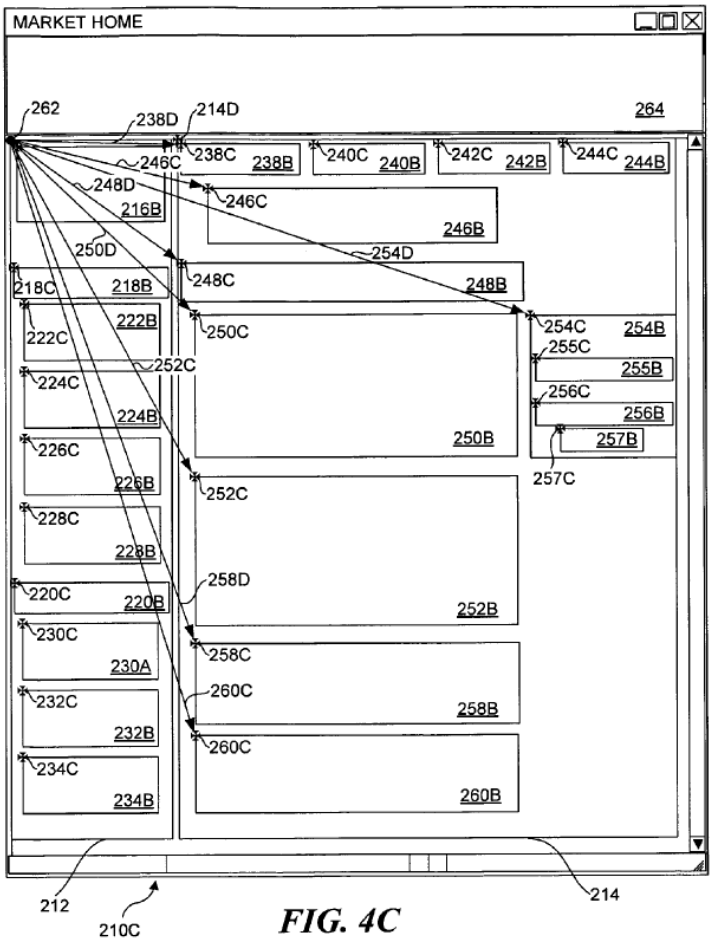


FIG. 4C

215. Claim 13 of the '154 patent is directed to a technological solution to a technological problem, and was not well-known, routine, or conventional.

216. The advantages of the '154 patent's claimed technical solution are substantial. Web pages could now "be rapidly rendered, zoomed, and panned. Moreover, the rendered displays provide substantially the same or identical layout as the original Web page, enabling users to easily navigate to selected content and features on familiar Web pages." '353 Patent, Abstract. This meant that users could finally be presented with web pages that fit and/or were effectively usable on devices with smaller screens, different resolutions, or different aspect ratios.

217. With the claimed invention, the web page could be presented as it would be on a desktop computer (as most web pages were designed for), and then rapidly panned and zoomed to

facilitate use on devices with smaller screens, different resolutions, or different aspect ratios. A Web page that was unusable when presented with the fixed resolution approach could now be zoomed to suit the device display. Today, this functionality is pervasive on smartphone and tablet devices and taken for granted. But prior to the claimed invention, it did not exist.

218. In addition, the claimed invention allowed the look and feel of the Web page to be maintained, even when designed for fixed resolutions. This allowed content providers to continue branding their Web pages with fixed resolution designs. It also allowed users to access Web pages as they were used to seeing them on desktop computers suited to their fixed resolution design, which meant users knew where to locate items on the page based on its familiar look and feel. Thus, the claimed invention made web pages more convenient for users.

219. The claimed invention has an additional advantage: internet content providers can rely on the fixed resolution approach instead of designing numerous different versions of the same web page to suit the screen size, resolution, and aspect ratio of different devices. For example, a web page designed for a 16-inch desktop display could be effectively viewed and used on a small hand-held mobile phone. This is superior to the alternative approach of designing multiple versions of the same web page to suit different devices, which was time consuming, costly, and impractical for many Web content developers.

220. Defendants have directly infringed (literally and/or under the doctrine of equivalents) at least claim 13 of the ’154 patent in violation of 35 U.S.C. § 271 *et seq*., through use, including testing of the Accused products without authority or license of the Accused Products.

221. Claim 13 recites, "A method performed by a mobile hand-held device including a touch-sensitive display and an HTML rendering engine." To the extent the preamble is limiting, the Accused Products satisfy this requirement.

222. The Accused Products, such as the Samsung Galaxy S10, are mobile hand-held devices with a touch-sensitive display and an HTML rendering engine, such as the Blink rendering engine used by the Samsung Internet and Chrome browsers. The Accused Products also include WebView which utilizes Blink for rendering HTML documents in Android apps outside of a browser, such as those received via email.

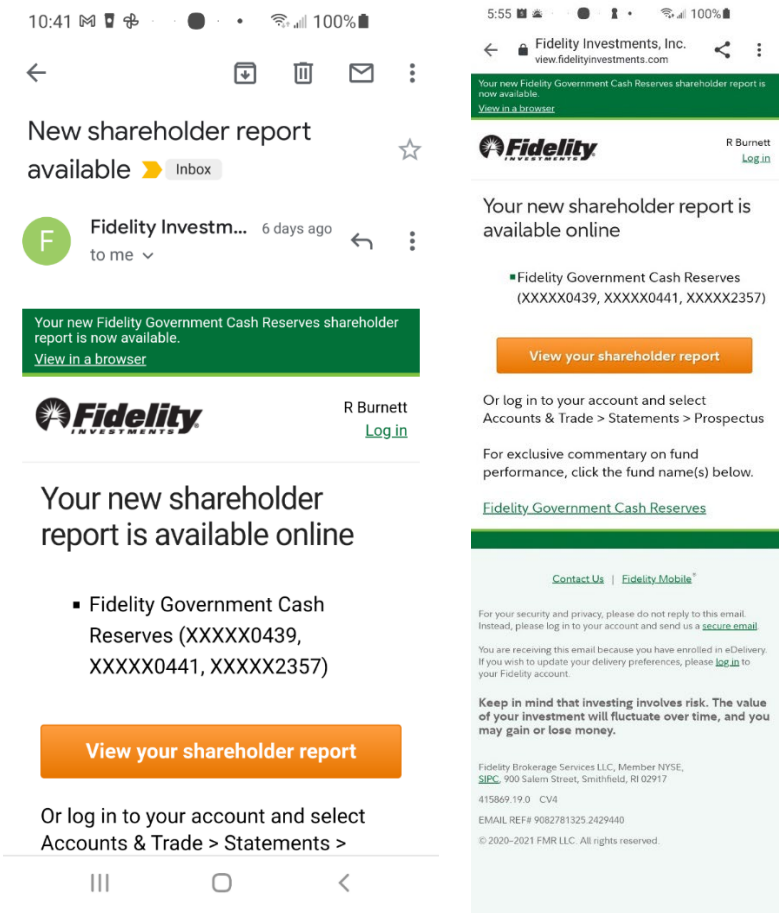


https://www.gsmarena.com/samsung_galaxy_s10-pictures-9536.php#.

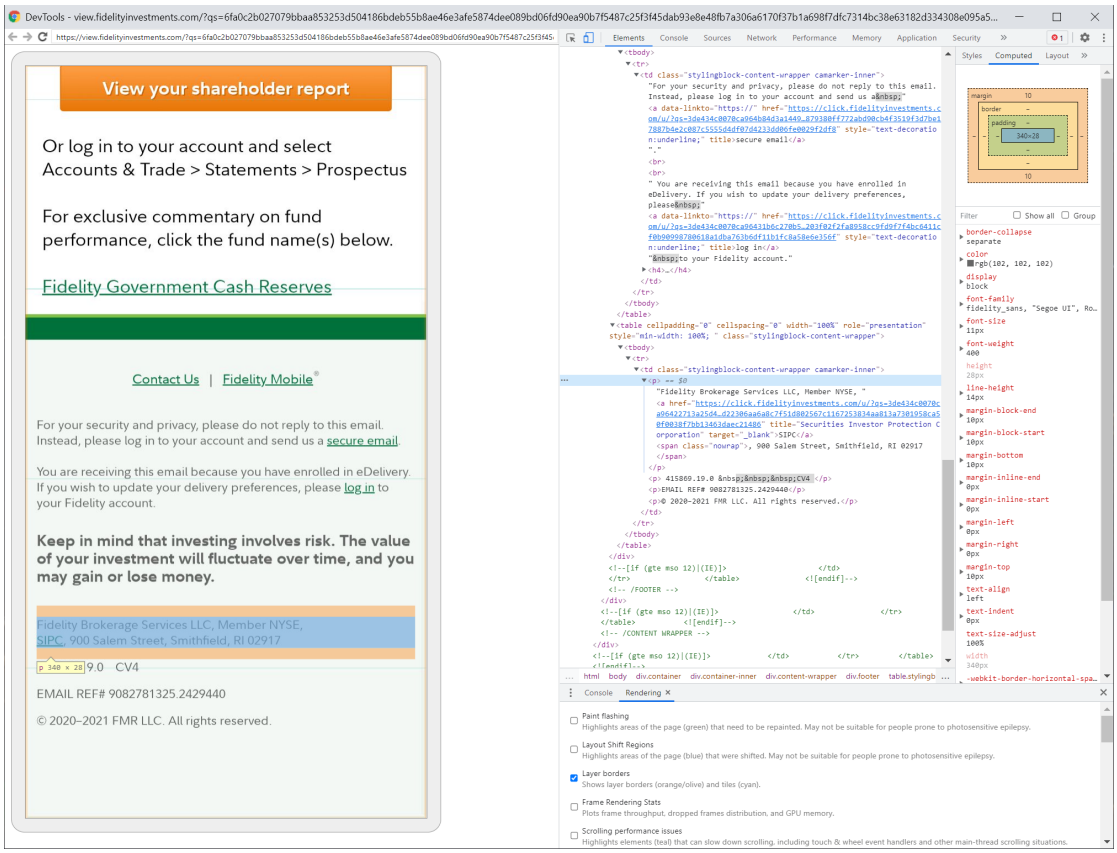
223. The Accused Products perform a method, which is described in further detail below.

224. Claim 13 of the '154 patent recites, "receiving an HTML document including HTML code and cascading style sheet (CSS) code and content associated with the HTML document, the HTML code comprising a plurality of HTML elements including at least one HTML paragraph element, at least one HTML image element, and at least one HTML hyperlink element." The Accused Products perform this operation.

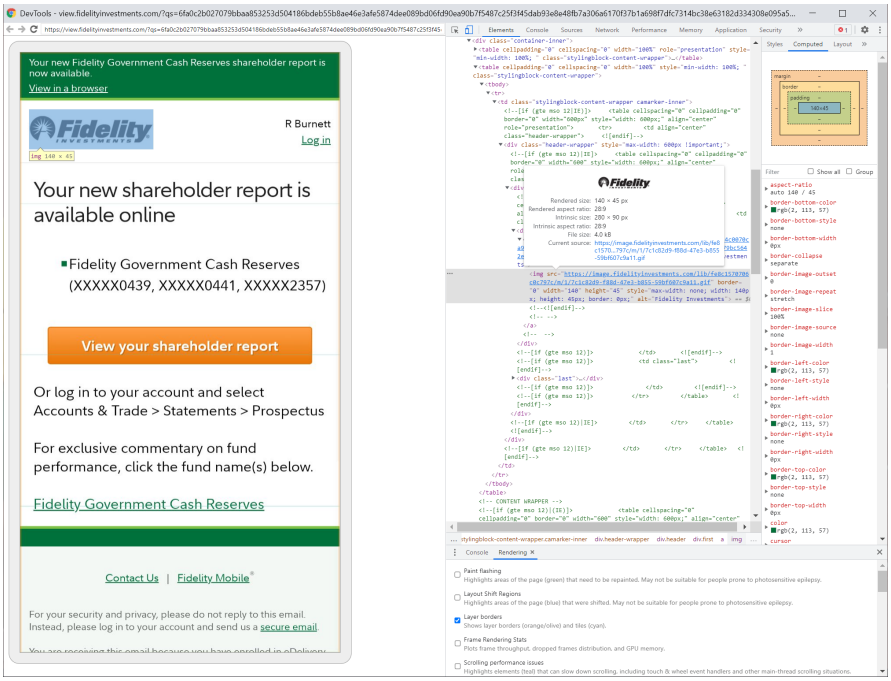
225. The Accused Products receive an HTML document including HTML code and content associated with the HTML document. An example is shown below, where a received HTML document is opened on the Galaxy S10. On the left is the image opened in the Gmail application. On the right is the HTML file opened in a Chrome browser after selecting an option to "View in a browser" in the Gmail application. The same URL is used to view the HTML document in the Gmail application and in a browser.



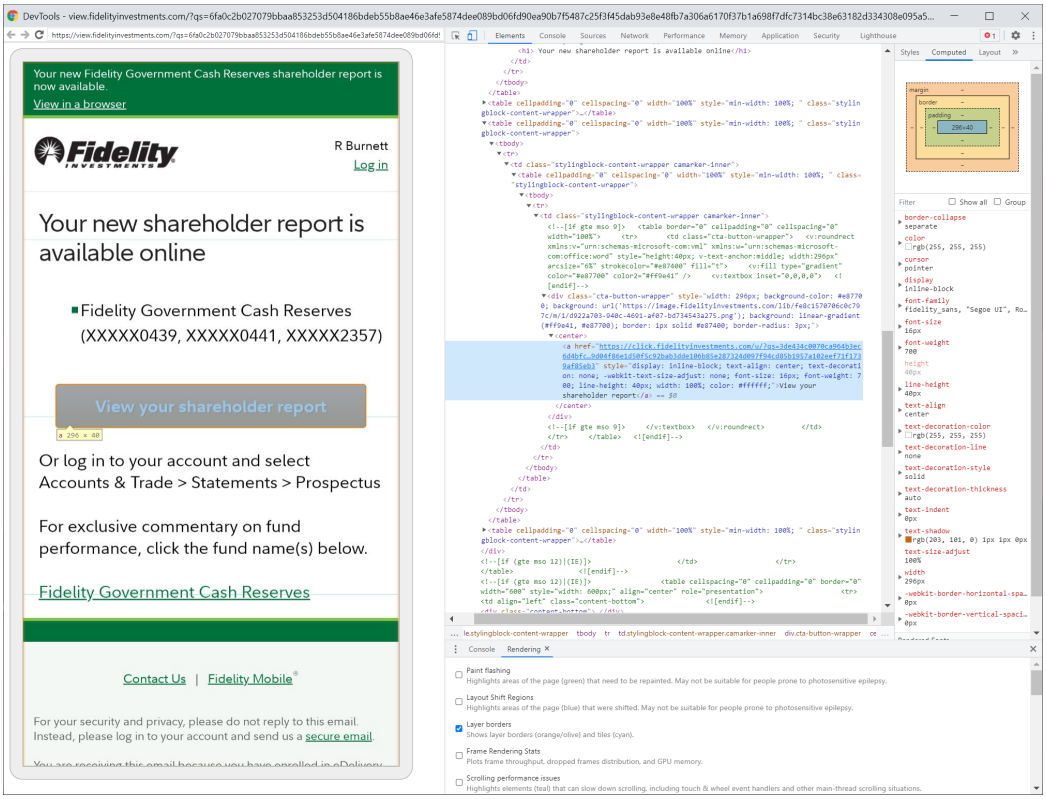
226. Below is the bounding box for an HTML paragraph element on the Web page as seen in the Chrome Web Inspector, denoted by the <P> tag.



227. Below is the bounding box for an HTML image element on the Web page as seen in the Chrome Web Inspector, denoted by the <img> tag.

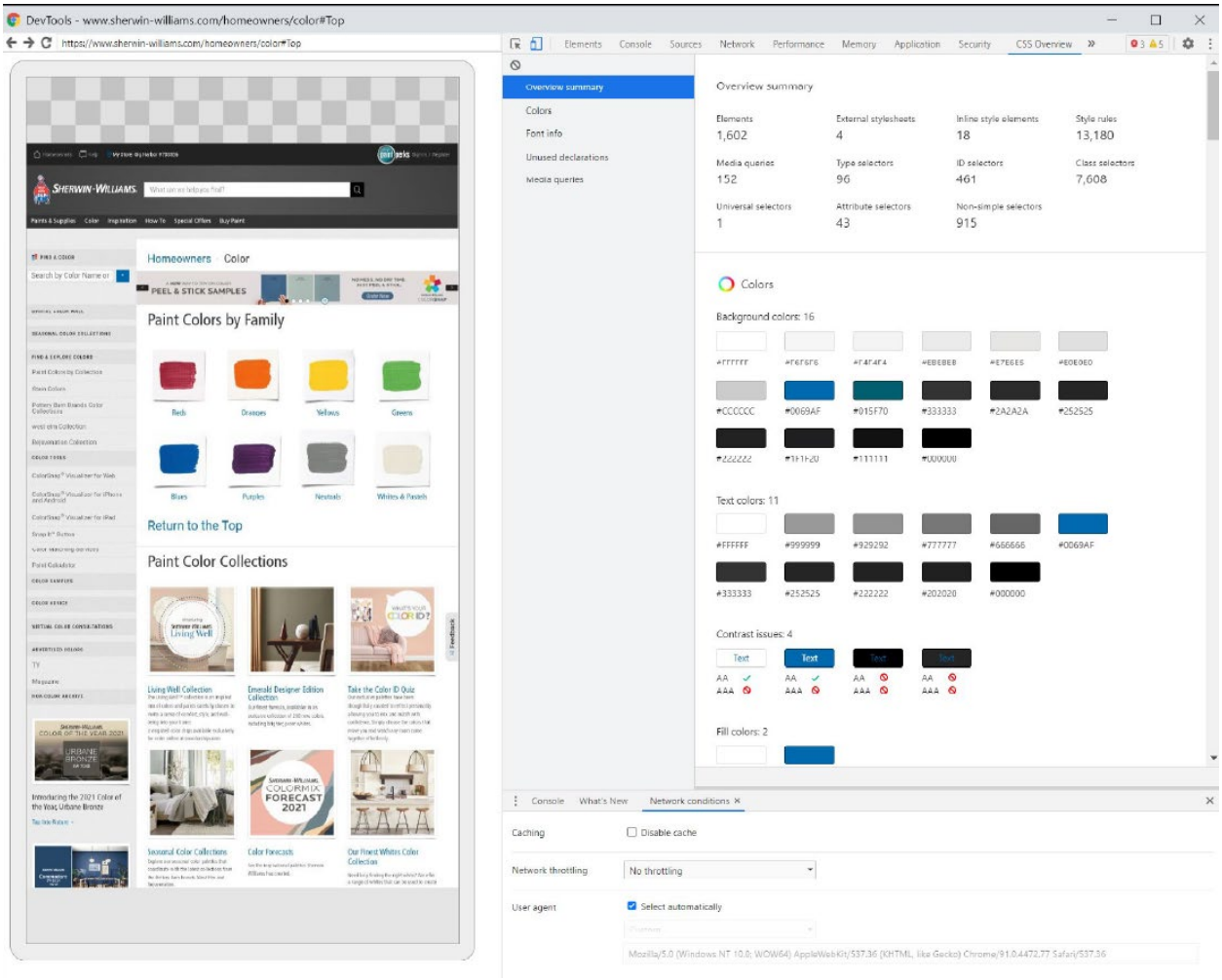


228. Below is the bounding box for an HTML hyperlink element on the Web page as seen in the Chrome Web Inspector, denoted by the <a href …> tag.

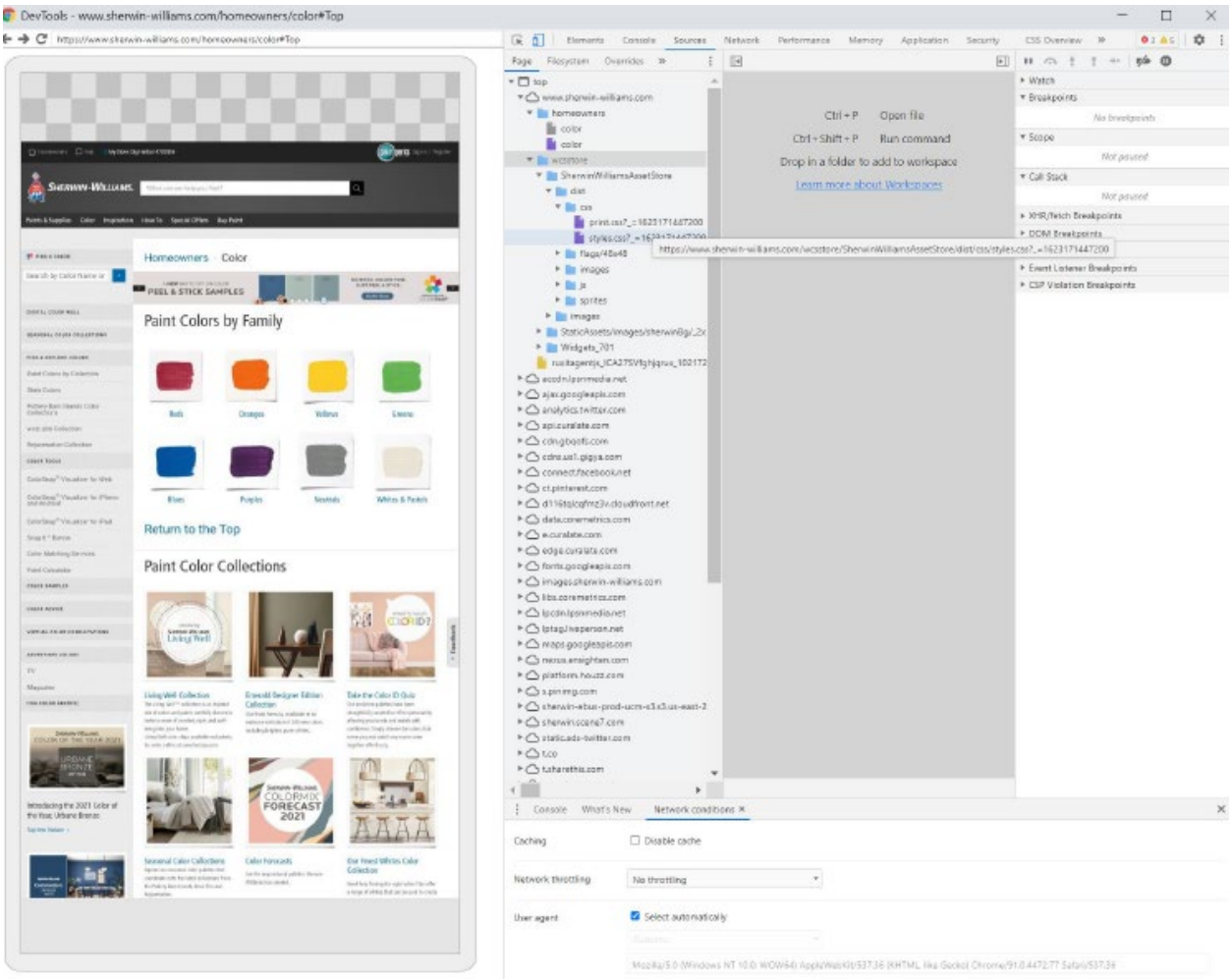


229. The received HTML document further includes cascading style sheet (CSS) code, including as shown by the code excerpts in the Chrome Web Inspector screenshots above.

230. The Chrome Web Inspector has a CSS overview tab that provides some information about CSS usage on the below page, which likewise includes at least one HTML paragraph element, at least one HTML image element, and at least one HTML hyperlink element.



231. The Sources tab of the Chrome Web Inspector shows that two CSS files were used for the Web page, including styles.css that is used for page layout and styling.



232. Claim 13 of the ’154 patent recites “processing the HTML document with the HTML rendering engine to render a first representation of the HTML document having an interpreted page layout, functionality, and design of the content associated with the HTML document that is in accordance with the HTML code and CSS code, wherein rendering the first representation of the HTML document includes, parsing the HTML document to identify the plurality of HTML elements; logically grouping content associated with HTML elements into HTML objects; generating page layout information including a bounding box for each HTML object; and storing information that links each HTML object with its corresponding page layout information, wherein the page layout information further includes information from which a page

layout location of each of the bounding boxes can be determined." The Accused Products perform these operations.

233. Claim 13 of the '154 patent recites "parsing the HTML document to identify the plurality of HTML elements."

234. Blink parses the HTML document to identify the plurality of HTML elements. Blink produces a document object model (DOM) tree where each DOM object corresponds to an HTML element. Blink also employs CSS rules and the CSS box model to generate a Layout Tree with page layout. Many of the core rendering processes employed by WebKit and Blink are similar as Blink and WebKit share the same core rendering code (WebCore). Blink uses a separate CSS layout engine called LayoutNG and uses a separate JavaScript engine. Aspects of the main flow for Blink are similar to the main flow for WebKit below, including use of HTML and CSS parsers and generation of a DOM and DOM Tree.

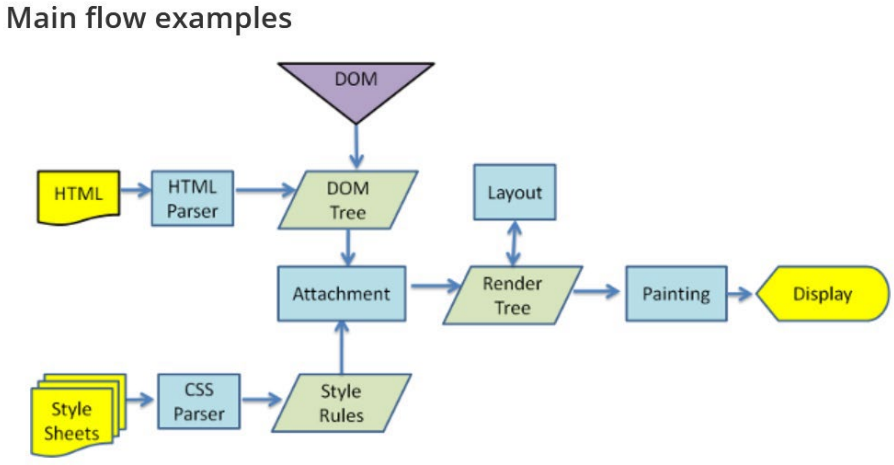


Figure : WebKit main flow

https://www.html5rocks.com/en/tutorials/internals/howbrowserswork/.

235. Claim 13 of the '154 patent recites "logically grouping content associated with HTML elements into HTML objects."

236. The Accused Products logically group content associated with HTML elements into HTML objects (DOM objects).

237. Claim 13 of the ’154 patent recites “generating page layout information including a bounding box for each HTML object.”

238. LayoutNG generates a Layout Tree with bounding boxes (called boxes for short) employing the CSS Box Model. There is a bounding box for each HTML object.

CSS Box model

The CSS box model describes the rectangular boxes that are generated for elements in the document tree and laid out according to the visual formatting model.
Each box has a content area (e.g. text, an image, etc.) and optional surrounding padding, border, and margin areas.

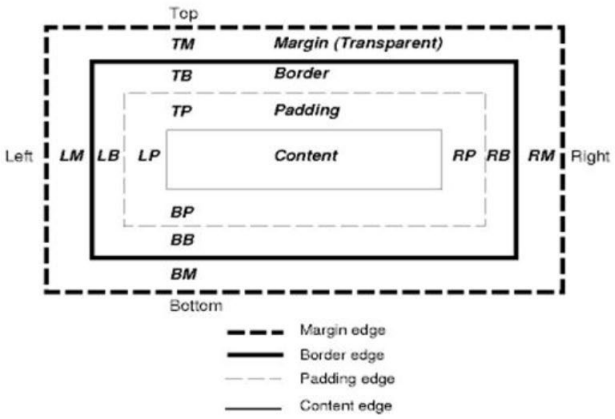


Figure : CSS2 box model

https://www.html5rocks.com/en/tutorials/internals/howbrowserswork/.

239. LayoutNG generates a Layout Tree comprising NGFragment and NGText Nodes.

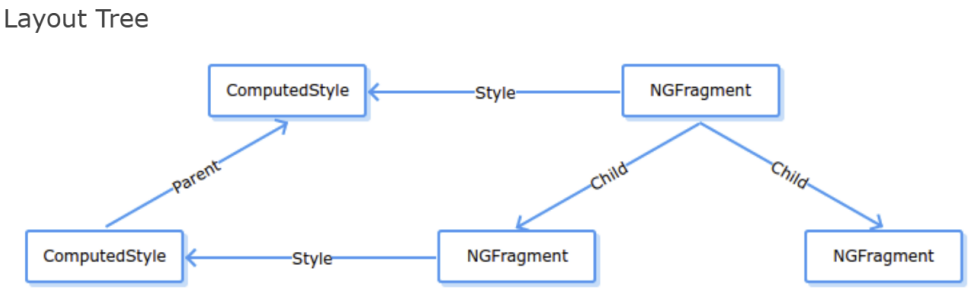


Figure shows the style tree (left) and layout tree (right) and their relationship.

The layout tree is made up entirely of NGFragment and NGText nodes and is immutable. Each node includes a back pointer to the computed style, a list of children and the size and location of the node itself. The location is relative to the immediate parent.

Given the immutability of the tree, use of relative coordinates, and the lack of a parent pointer entire subtrees may be reused across layouts. This concept is discussed in detail in the subsequent layouts section.

https://developer.chrome.com/docs/chromium/blinkng.

240. The Layout Tree also represents the bounding box hierarchy of the HTML document.

Pipelining style, layout, and pre-paint

Collectively, the rendering phases before *paint* are responsible for the following:

- Running the *style cascade* algorithm to calculate final style properties for DOM nodes.
- Generating the layout tree representing the box hierarchy of the document.
- Determining size and position information for all boxes.
- Rounding or snapping sub-pixel geometry to whole pixel boundaries for painting.
- Determining the properties of composited layers (affine transformation, filters, opacity, or anything else that can be GPU accelerated).
- Determining what content has changed since the previous paint phase, and needs to be painted or repainted (paint invalidation).

https://developer.chrome.com/docs/chromium/blinkng.

241. LayoutNG uses ClientRects to size and position DOM objects relative to the viewport.

ClientRect	Size and position of an Element relative to the viewport.

Walking up the Tree

ClientRect calculations and IntersectionObserver need to perform a tree-walk up to the root in order to apply transform/layout information. Transform information is not on the layout tree. Adding layout-parent pointers in the Layout Tree would mean that we lose the immutability property; so instead the parent-layout pointers will be stored on the DOM tree. These operations will begin at the DOM node, ask its NGUnplacedFragment for layout information, then walk up the DOM tree to its parent-layout and so forth.

https://docs.google.com/document/d/1uxbDh4uONFQOiGuiumlJBLGgO4KDWB8ZEkp7Rd47fw4/edit?tab=t.0#heading=h.guvbepjyp0oj.

242. Claim 13 of the '154 patent recites "storing information that links each HTML object with its corresponding page layout information, wherein the page layout information further includes information from which a page layout location of each of the bounding boxes can be determined." The Accused Products perform this operation.

243. The Blink rendering engine performs a Layout stage to determine the size and position of DOM elements (HTML objects) on the screen. A layout tree is generated representing the (bounding) box hierarchy of the document, where each box is associated with a DOM node (DOM elements in the DOM tree are also called DOM nodes). The layout information comprising the location and size of the box for each DOM element is linked with the DOM element, thereby storing information that links each HTML object with its corresponding page layout information, wherein the page layout information further includes information from which a page layout location of each of the bounding boxes can be determined.

The stages are:

1. *Animate:* change computed styles and mutate *property trees* over time based on declarative timelines.
2. *Style:* apply CSS to the DOM, and create *computed styles*.
3. *Layout:* determine the size and position of DOM elements on the screen, and create the *immutable fragment tree*.
4. *Pre-paint:* compute property trees and invalidate any existing *display lists* and GPU *texture tiles* as appropriate.
5. *Scroll:* update the scroll offset of documents and scrollable DOM elements, by mutating property trees.
6. *Paint:* compute a display list that describes how to raster GPU texture tiles from the DOM.
7. *Commit:* copy property trees and the display list to the compositor thread.
8. *Layerize:* break up the display list into a *composited layer list* for independent rasterization and animation.
9. *Raster, decode and paint worklets:* turn display lists, encoded images, and paint worklet code, respectively, into GPU texture tiles.
10. *Activate:* create a *compositor frame* representing how to draw and position GPU tiles to the screen, together with any visual effects.
11. *Aggregate:* combine compositor frames from all the visible compositor frames into a single, global compositor frame.
12. *Draw:* execute the aggregated compositor frame on the GPU to create pixels on-screen.

https://developer.chrome.com/docs/chromium/renderingng-architecture#rendering_pipeline_structure.

Pipelining style, layout, and pre-paint

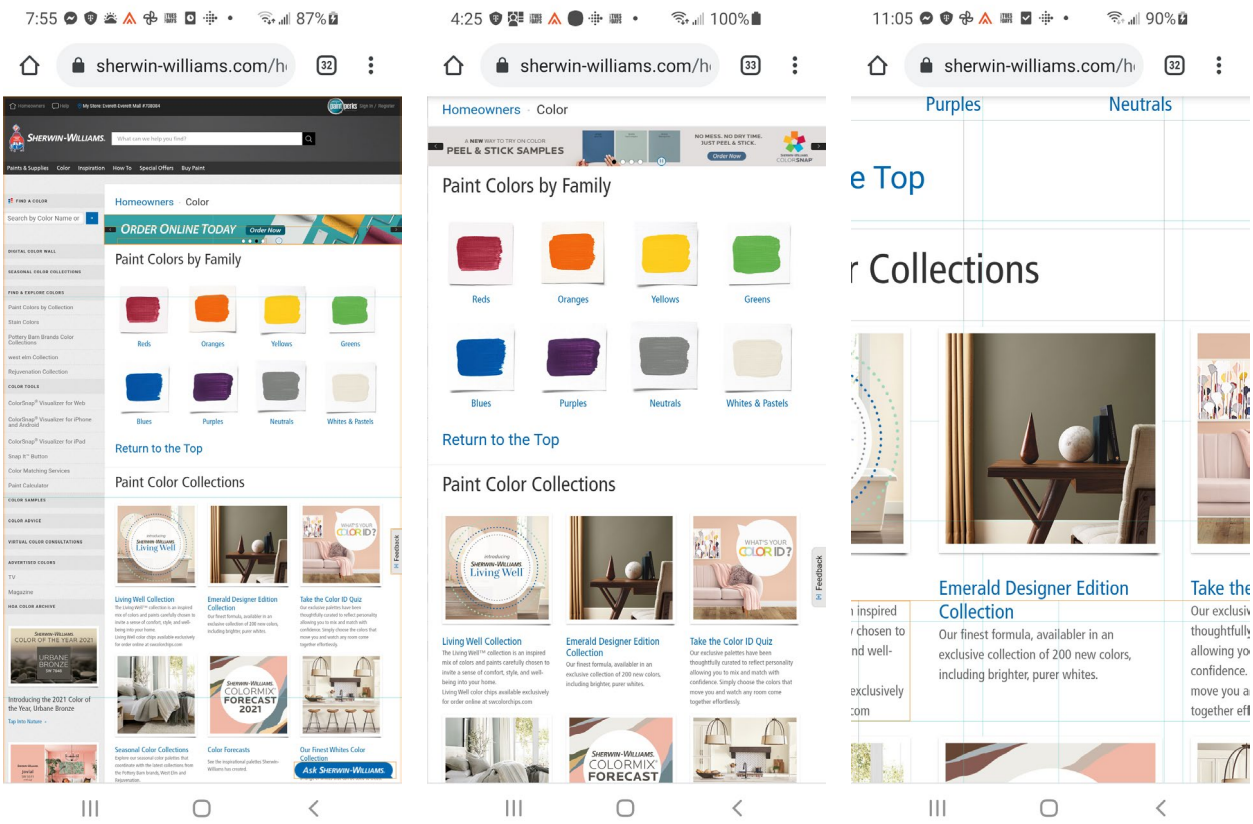
Collectively, the rendering phases before *paint* are responsible for the following:

- Running the *style cascade* algorithm to calculate final style properties for DOM nodes.
- Generating the layout tree representing the box hierarchy of the document.
- Determining size and position information for all boxes.
- Rounding or snapping sub-pixel geometry to whole pixel boundaries for painting.
- Determining the properties of composited layers (affine transformation, filters, opacity, or anything else that can be GPU accelerated).
- Determining what content has changed since the previous paint phase, and needs to be painted or repainted (paint invalidation).

https://developer.chrome.com/docs/chromium/blinkng.

244. Claim 13 of the '154 patent recites "translating the first representation of the HTML document to generate a scalable vector representation of the HTML document." The Accused Products perform this operation.

245. The Samsung Internet and Chrome browsers, and the Gmail application, of the Accused Products translate the first representation of the HTML document to generate a scalable vector representation of the HTML document. The screenshots below demonstrate zooming in on the scalable vector representation.



246. WebView (and the Samsung Internet and Chrome browsers) creates a vector representation of the entire page, which is scalable.

Picture

- Vector representation of the entire page (not just the visible area)
- No need to go back to webkit when:
  - Scrolling
  - Zooming

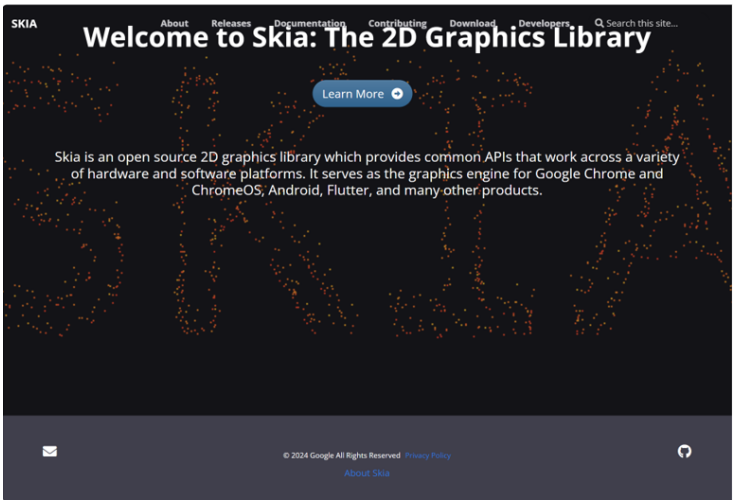
Friday, June 29, 2012

19

19

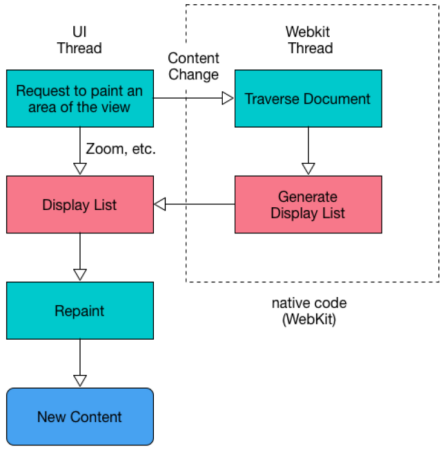
Android WebView — Google IO 2012.

247. The Samsung Internet and Chrome browsers use the Skia 2D vector graphics rendering software/library, which is also referred to as the Skia Graphics Engine.



248. WebView (and the Samsung Internet and Chrome browsers) records lists of painting instructions for the full page and builds display lists. Zooming is accomplished by changing scale and translation parameters in a transformation matrix (SkMatrix) to scale and/or translate position vector coordinates and calling the painting instructions in the display list. While the following describes the approach used by the Chrome browser with WebKit, the current painting process using Blink is similar.

So, instead of directly asking webkit to *paint* the screen, the team had a better idea. They decided to modify Webkit to *record* the list of painting instructions for the full page, but to not paint directly — basically using a "display list" model. The list of painting commands were for example "draw this text here", or "draw this image there".



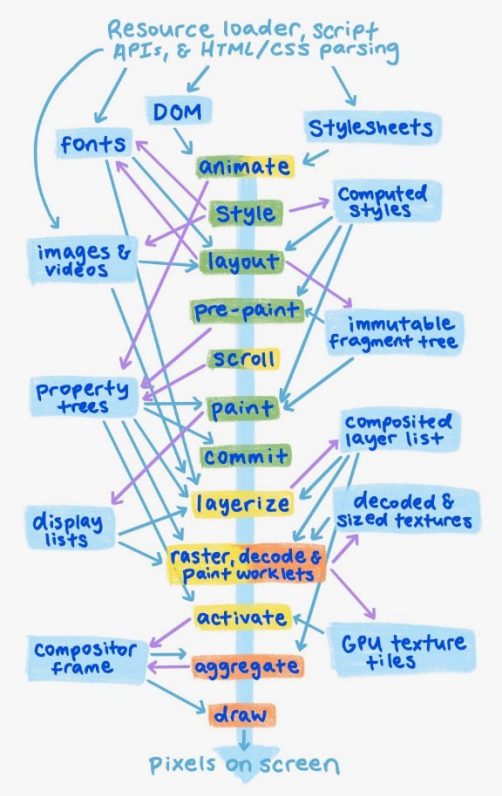
Rendering loop (Android)

When the user would need to zoom in for example, it was not necessary to ask Webkit to repaint the screen — the webview instead could just replay the already present display list, only changing the zoom factor. This also gave a rather nice side-effect that the screen was always sharp — no text blurriness when zooming in, no missing content, as the content was always being redrawn "the right way".

You could think of this display list as a way to have a vector representation of the entire page, that you could scale on demand and redraw — the difference between illustrator and photoshop.

https://medium.com/@camaelon/what-s-in-a-web-browser-83793b51df6c.

249. Display lists are used in the current Rendering Pipeline.



The stages are:

1. *Animate:* change computed styles and mutate *property trees* over time based on declarative timelines.
2. *Style:* apply CSS to the DOM, and create *computed styles*.
3. *Layout:* determine the size and position of DOM elements on the screen, and create the *immutable fragment tree*.
4. *Pre-paint:* compute property trees and invalidate any existing *display lists* and GPU *texture tiles* as appropriate.
5. *Scroll:* update the scroll offset of documents and scrollable DOM elements, by mutating property trees.
6. *Paint:* compute a display list that describes how to raster GPU texture tiles from the DOM.
7. *Commit:* copy property trees and the display list to the compositor thread.
8. *Layerize:* break up the display list into a *composited layer list* for independent rasterization and animation.
9. *Raster, decode and paint worklets:* turn display lists, encoded images, and paint worklet code, respectively, into GPU texture tiles.
10. *Activate:* create a *compositor frame* representing how to draw and position GPU tiles to the screen, together with any visual effects.
11. *Aggregate:* combine compositor frames from all the visible compositor frames into a single, global compositor frame.
12. *Draw:* execute the aggregated compositor frame on the GPU to create pixels on-screen.

https://developer.chrome.com/docs/chromium/renderingng-architecture#pipeline.

Display lists and paint chunks

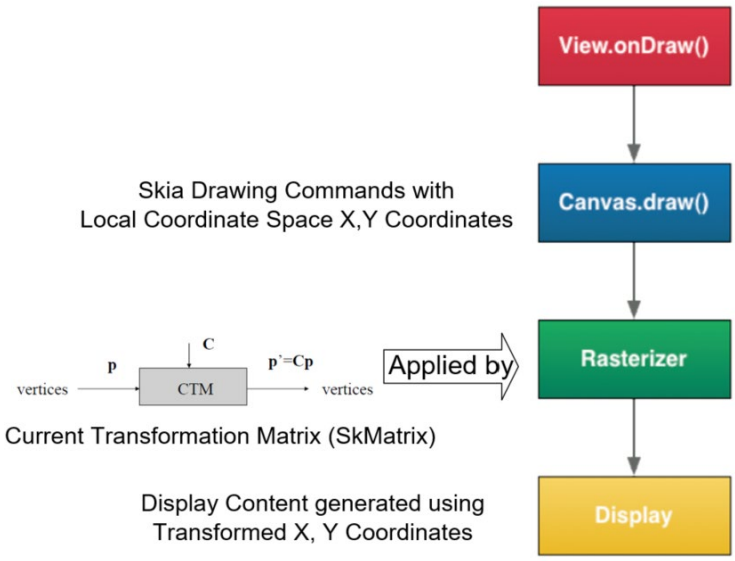
A display item contains low-level drawing commands (see here) that can be rasterized with Skia. Display items are typically simple, with just a few drawing commands, such as drawing a border or background. The paint tree walk iterates over the layout tree and associated fragments following CSS painting order to produce a display item list.

For example:



https://developer.chrome.com/docs/chromium/renderingng-data-structures#display_lists_and_paint_chunks.

250. Skia uses a Current Transformation Matrix (SkMatrix) for scaling and translation of content. An example is shown below for Skia painting.



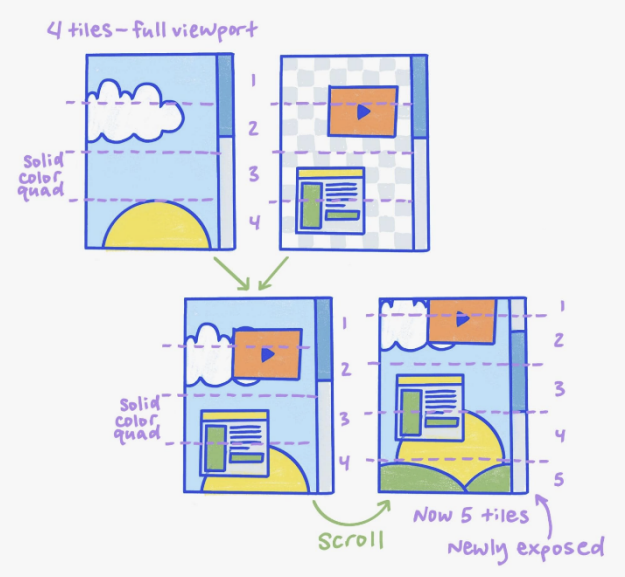
Android Accelerated Rendering Google IO 2011 (modified).

251. The rendering pipeline uses the Skia display lists to rasterize pixels onto GPU texture tiles.

Tiles

In theory, a render process or browser process compositor could rasterize pixels into a single texture the full size of the renderer viewport and submit that texture to Viz. To display it, the display compositor would just have to copy the pixels from that single texture into the appropriate position in the frame buffer (for example, the screen). However, if that compositor wanted to update even a single pixel it would need to re-rasterize the full viewport and submit a new texture to Viz.

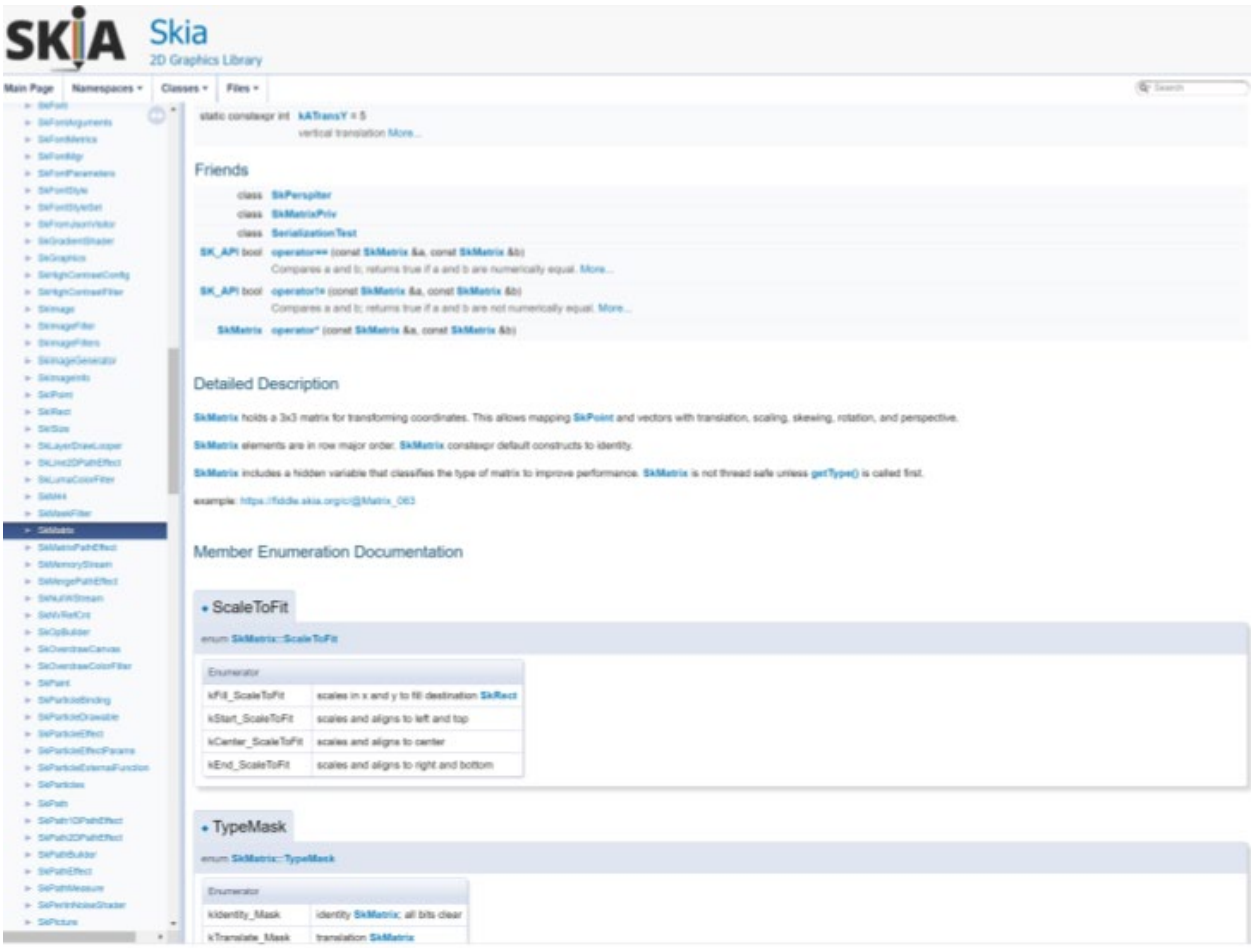
Instead, the viewport is divided into tiles. A separate GPU texture tile backs each tile with the rasterized pixels for part of the viewport. The renderer can then update individual tiles or even just change the position on screen for the existing tiles. For example, when scrolling a website, the position of existing tiles would shift up and only occasionally a new tile would need to be rasterized for content further down the page.



This image depicts an image of a sunny day, with four tiles. When a scroll occurs, a fifth tile begins to appear. One of the tiles happens to only have one color (sky blue), and there is a video and an iframe on top.

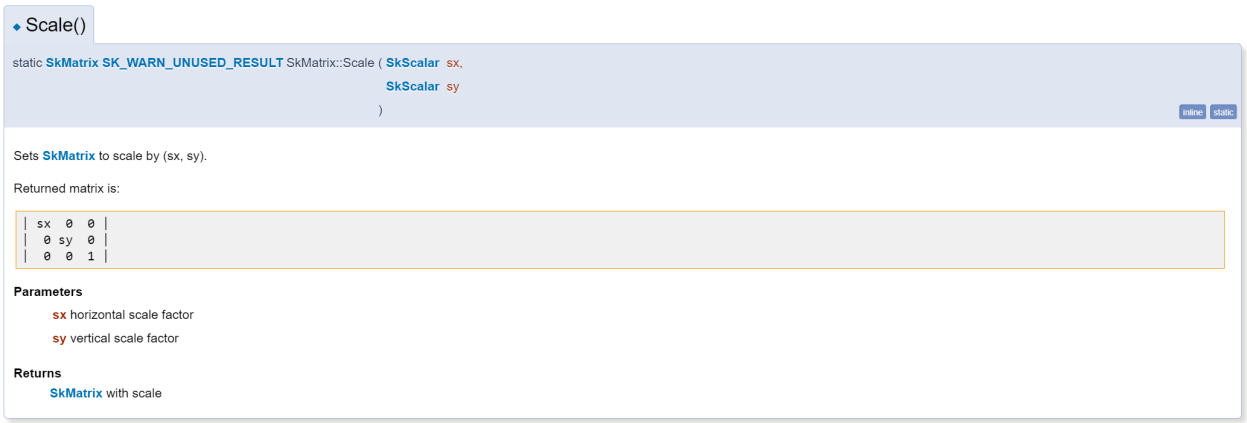
https://developer.chrome.com/docs/chromium/renderingng-data-structures#tiles.

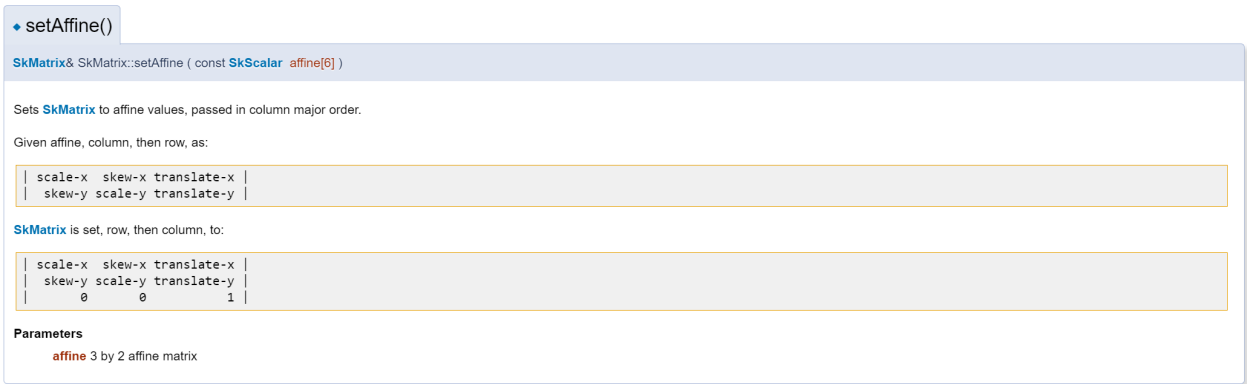
252. The Display List uses the SKMatrix class to effect scaling. SKMatrix holds a 3x3 matrix for transforming coordinates (also known as an affine transformation). SKMatrix is used to effect translation and scaling. Skia renders images by ScaleToFit the destination rectangle (SKRect). SKRect is scaled using the current SKMatrix, and then the Image is scaled to fit the SKRect that is generated.



https://api.skia.org/classSkMatrix.html.

253. The Skia Scale() function can be used to change the sx (horizontal) and sy (vertical) scale factors in the SKMatrix. The setAffine() function can be used to set values for the first two rows of the SKMatrix. This enables scaling and translation to be performed together.





254. Claim 13 of the '154 patent recites "wherein generating the scalable vector representation includes, defining a primary datum corresponding to the interpreted page layout; for each HTML object, defining an object datum corresponding to a layout location datum for the HTML object's bounding box; generating a vector from the primary datum to the object datum for the HTML object's bounding box; and creating a reference that links the HTML object to the vector that is generated." The Accused Products perform these operations.

255. Claim 13 of the '154 patent recites "defining a primary datum corresponding to the interpreted page layout."

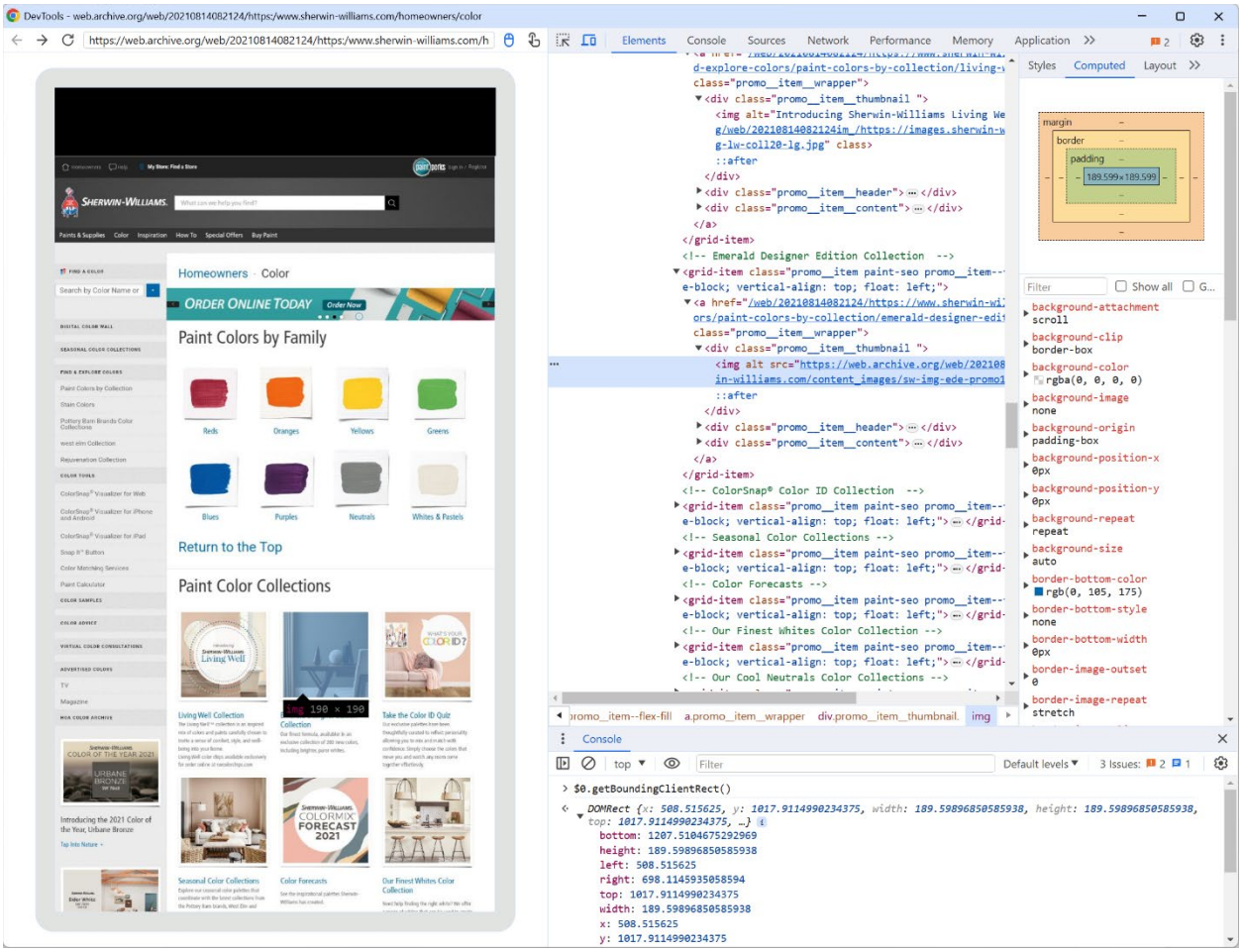
256. The primary datum corresponding to the interpreted page layout is the upper-left hand corner of the viewport, which is also the origin of the local coordinate space (local coordinate system). The origin of the local coordinate space has x,y coordinates of 0,0.

257. Claim 13 of the '154 patent recites "for each HTML object, defining an object datum corresponding to a layout location datum for the HTML object's bounding box."

258. Each ClientRect has a defined object datum corresponding to the layout location datum for an HTML object's bounding box relative to the viewport origin (0,0). The Blink Rendering engine stores the ClientRect bounding box and DOMRect for each element in the DOM that has associated display content. Each object has an associated x, y coordinate comprising the object datum for the object.

ClientRect	Size and position of an Element relative to the viewport.

259. The object datum for the image highlighted in the screenshot below is 508.515 (x), 1017.911 (y) in LayoutUnits.

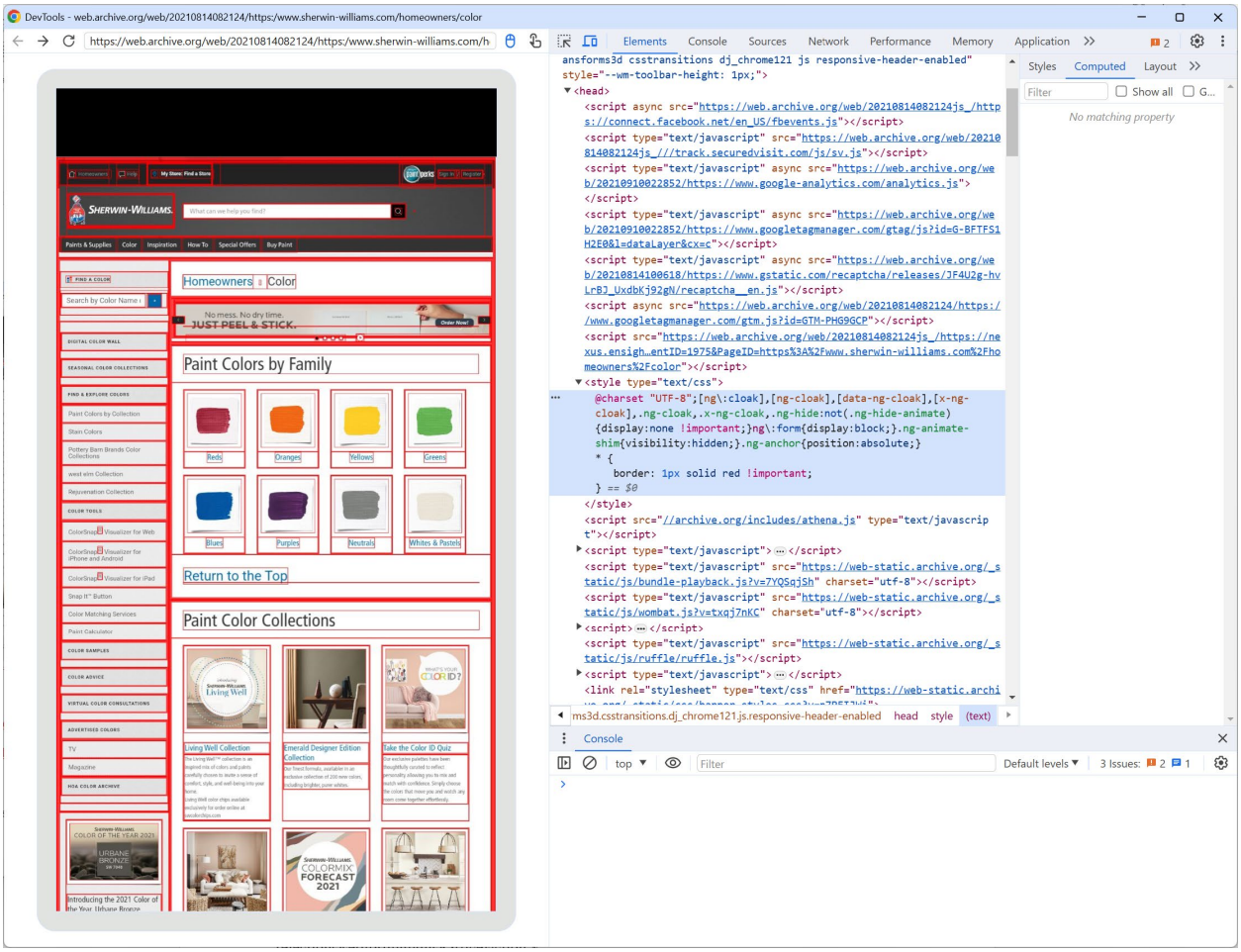


260. Claim 13 of the ’154 patent recites “generating a vector from the primary datum to the object datum for the HTML object's bounding box” and “creating a reference that links the HTML object to the vector that is generated.” The Accused Products perform these operations by storing the x, y coordinates of the DOMRect for the HTML object as a position vector in a local coordinate space comprising a vector space.

261. The object datums comprise x, y coordinates in the local coordinate space; each x, y coordinate in the local coordinate space is a position vector, with the tail of each position vector

at the 0,0 origin (the primary datum) and the head of the position vector at the x, y coordinate (the object datum). The object datums (x, y coordinates) are used to support a vector representation of the Web page that can be scaled and translated to enable the user to zoom in and out views of the Web page with perfect scaling.

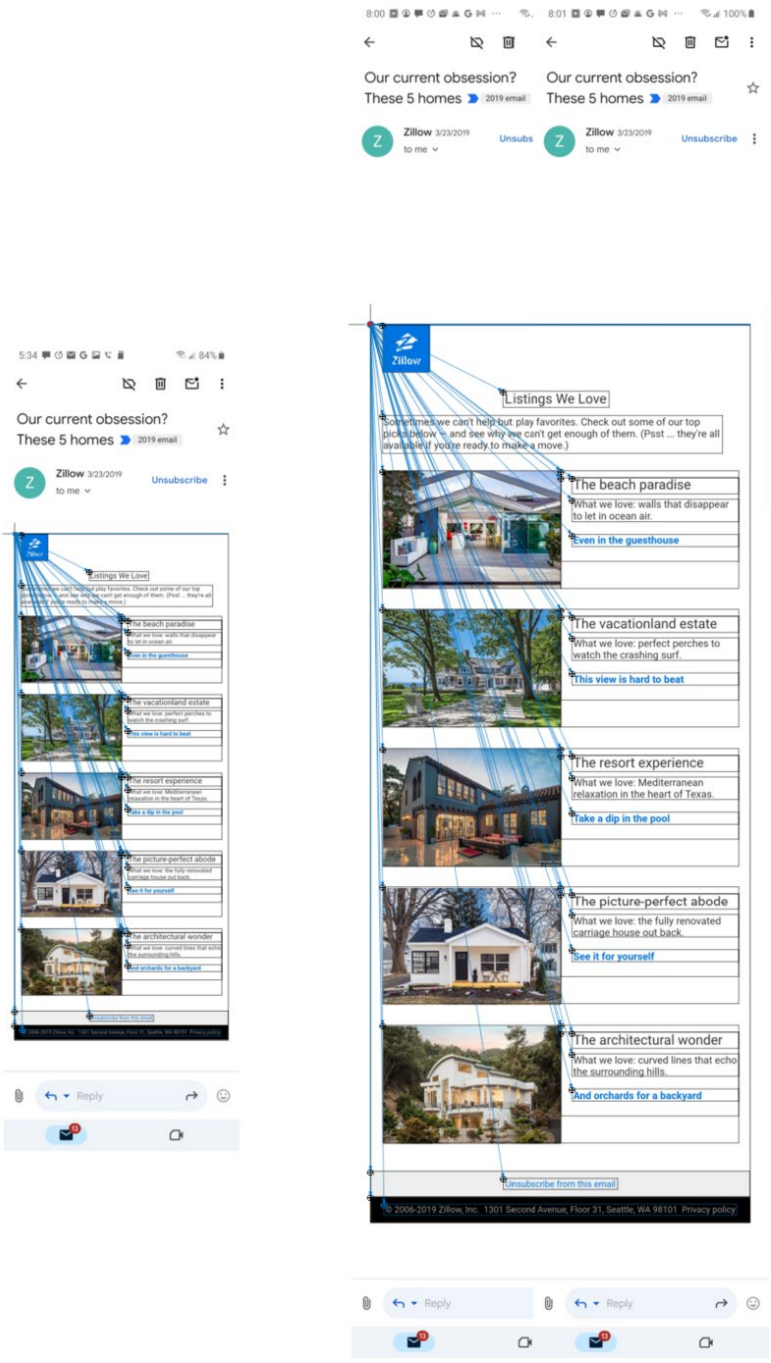
262. A visual representation of the Layout Tree structure can be shown by modifying the HTML document to render a visual border for each bounding box.



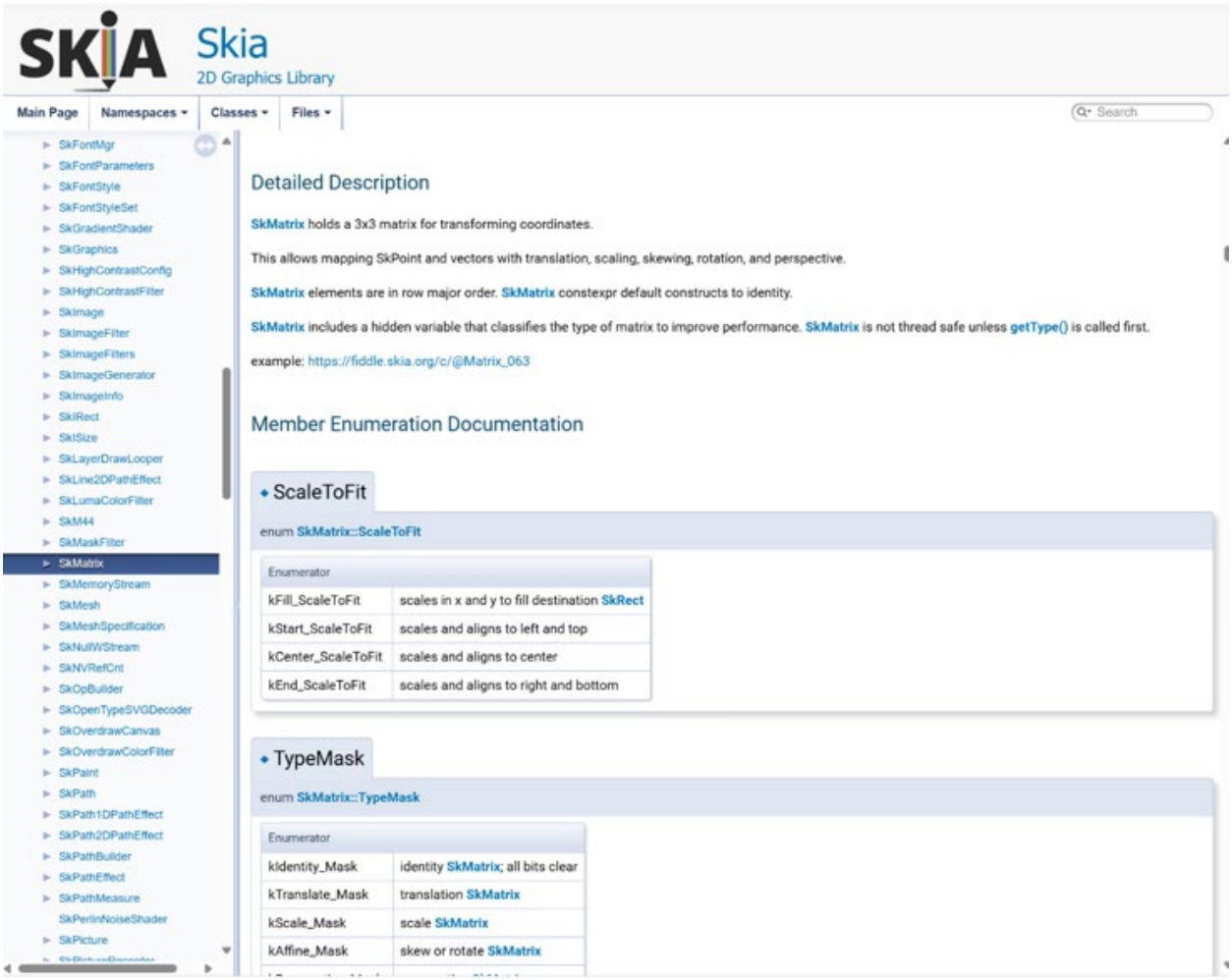
The following code is added to the end of the <style> section highlighted above:

```
* {
  border: 1px solid red !important;
}
```

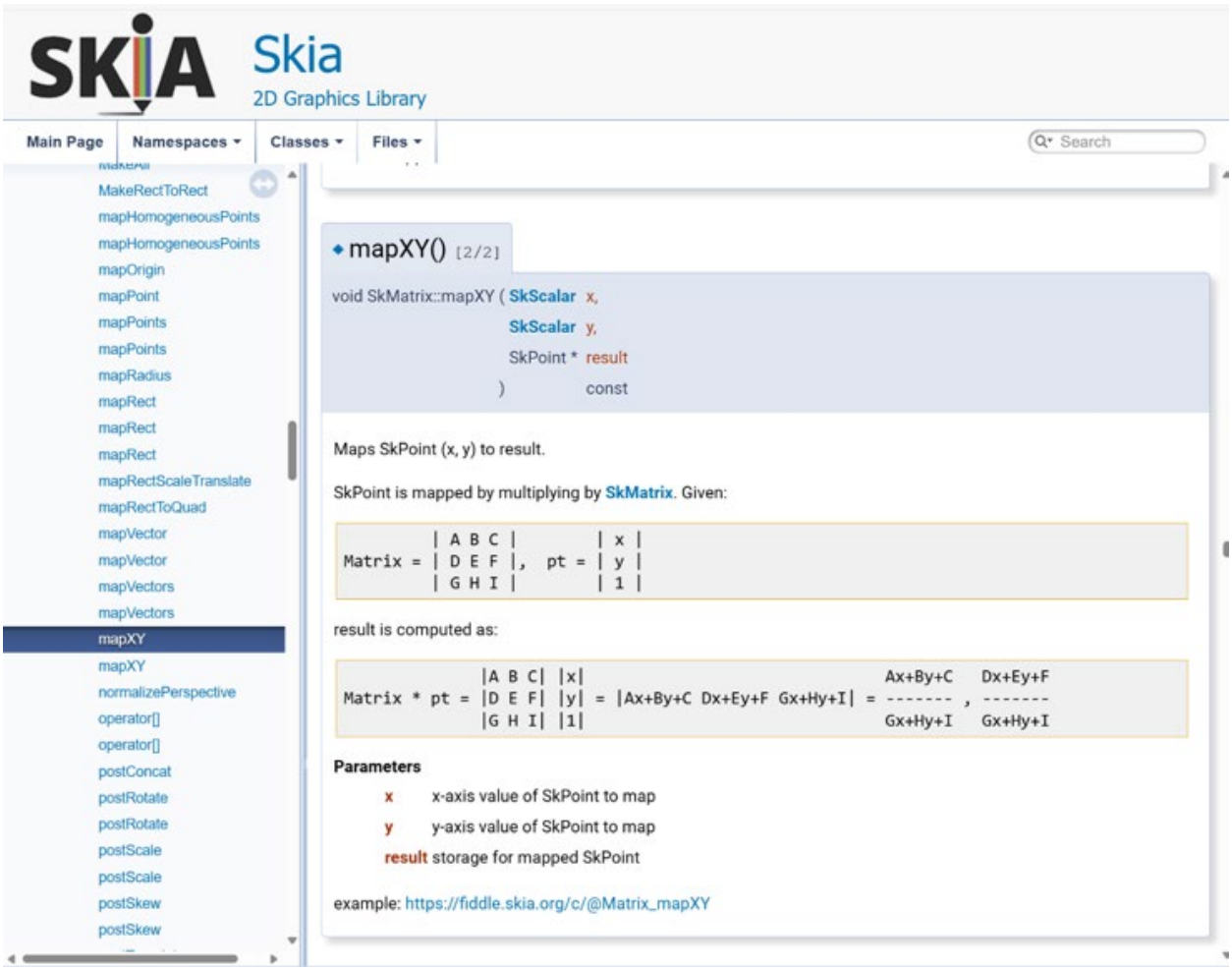
263. The following illustrates vector-based scaling of the page layout for a Web page using the Android Chrome browser. This example shows vectors for selected content. The tail of each vector is at the page, viewport, and local coordinate space origin (0, 0) and the head of each vector is at the datum for an associated object.



264. Skia uses vector mathematics to scale and translate the object datums (comprising SkPoints with x, y coordinates) to support zooming and panning (scrolling) and to rasterize tiles. This utilizes the SkMatrix class.

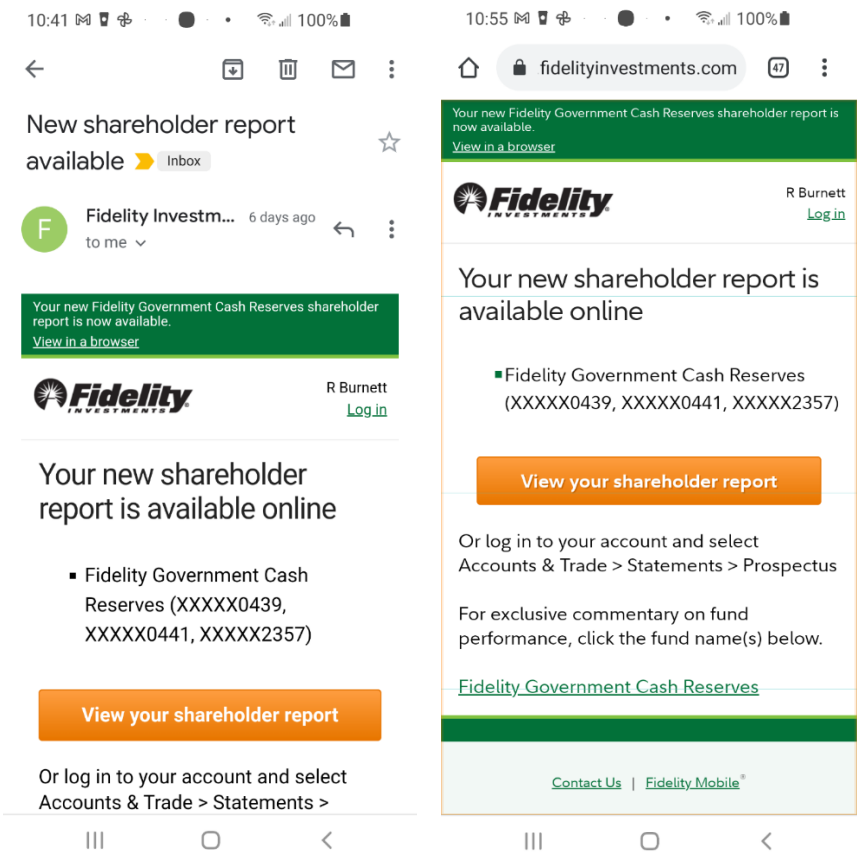


265. Skia provides multiple functions (methods) to scale the coordinates of points (position vectors) in the local coordinate space, such as the mapXY() method. The point is an SkPoint. SkScalars x and y are the x and y scale factors, respectively.



266. Claim 13 of the '154 patent recites "render the scalable vector representation of the HTML document on the touch-sensitive display using a first scale factor to display the HTML document at a first zoom level under which the HTML document is displayed to fit across a width of the touch-sensitive display." The Accused Products perform this operation.

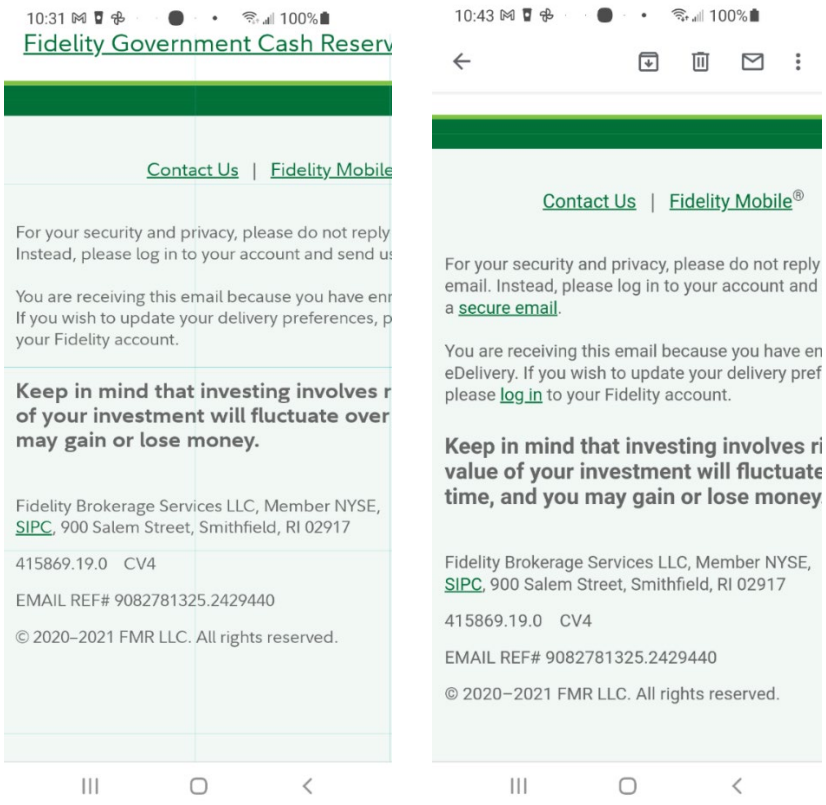
267. The screenshots below show a scalable vector representation of an HTML document on the touch-sensitive display of a Galaxy S10. A first scale factor is applied to display the HTML document at a first zoom level under which the HTML document is displayed to fit across a width of the touch-sensitive display. The first image was taken in the Gmail application. The second image was taken in the Chrome browser after selecting the "View in a browser" option in the Gmail application.



268. Claim 13 of the '154 patent recites "wherein the scalable vector representation of the HTML document is configured to enable a user to view the HTML document at one or more user-defined zoom levels by rendering the scalable vector representation on the touch-sensitive display using one or more respective scale factors in response to associated user inputs made via the touch-sensitive display, and wherein the interpreted page layout, functionality, and design of the content associated with the HTML document is preserved at each of the first zoom level and the one or more user-defined zoom levels." The Accused Products satisfy these requirements.

269. Below are screenshots zoomed in on the pages above. The first image was taken in the Chrome browser. The second was taken in the Gmail application. Zooming was done by pinch zooming on the Galaxy S10's touch-sensitive display. As can be seen, the HTML document is viewed at a different scale factor due to the user-defined zoom level. The interpreted page

layout, functionality, and design of the content associated with the HTML document is preserved at both zoom levels.



Samsung's Inducement of Infringement of the '154 Patent

270. SoftView incorporates by reference and re-alleges the foregoing paragraphs of this Complaint as if fully set forth herein.

271. The '154 patent was filed on November 18, 2016 and issued on September 25, 2018. On information and belief, Defendants had knowledge of the '154 patent on or soon after its issuance date.

272. Defendants were served on October 5, 2011, with the Second Amended Complaint in the Delaware litigation alleging infringement of the '353 and '926 patents. The '154 patent is in the same family as the '353 and '926 patents.

273. Defendants had knowledge of the ’154 patent or should have known of the ‘154 patent in consideration that Defendants were in prior litigation involving the ‘353 and ‘926 patents which are members of the same patent family.

274. In addition, as alleged with respect to the ’729 patent, Defendants were aware of the ’729 patent through it being cited and/or discussed in applications of Defendant SEA. The ’154 patent is a continuation of the ’729 patent, and the ’729 patent was pending during the Delaware litigation against Defendants.

275. SoftView is informed and believes, and thereon alleges, that Defendants have actively induced infringement of at least claim 13 of the ’154 patent in violation of 35 U.S.C. § 271(b) by knowingly and intentionally encouraging or aiding third parties to use infringing software and hardware products in this judicial district and elsewhere throughout the United States, without license or authority from SoftView, including at least the Accused Products. SoftView is informed and believes, and thereon alleges, that these third parties have infringed at least claim 13 of the ’154 patent in violation of 35 U.S.C. § 271(a) by using infringing software and hardware products, including the Accused Products. The Defendants through at least their user manuals, Web site support and training materials actively induced their customers and users of the Accused Products to infringe the ’154 patent. Furthermore, the natural and intended use of the Accused Products (*e.g.*, browsing the internet with Samsung Internet and Chrome browsers and viewing HTML email with the Gmail and Android Mail apps) infringes at least claim 13 of the ’154 patent. Defendants know and intend that the Accused Products are used in this infringing manner. In addition, SoftView reasserts evidence presented in Count I *supra* supporting that Defendants knowingly induce infringement of at least claim 13 of the ’154 patent.

276. Defendants have induced infringement of at least claim 13 of the ’154 patent.

277. On information and belief, Defendants have committed acts of infringement notwithstanding actual knowledge of the ’154 patent and without a good faith basis to believe that their activities do not infringe at least claim 13 of the ’154 patent. All infringement of the ’154 patent following Defendants’ knowledge of the ’154 patent is willful, and SoftView is entitled to treble damages and attorneys’ fees and costs incurred in this action under 35 U.S.C. §§ 284 and 285.

278. In the event Defendants assert they were not aware of the ’154 patent, or lack the requisite knowledge and intent to induce infringement, Defendants would have been willfully blind of SoftView’s patent rights.

279. In view of the Delaware litigation and multiple patent applications pending and that issued during the litigation from the same family as the ’353 and ’926 patents, along with the citation of the related ’729 patent to Defendant SEA, Defendants were aware of a high probability of the existence of such applications and patents and infringement, and/or would had to have taken deliberate actions to avoid learning of the existence of such applications and patents and infringement, including the ’154 patent.

280. Defendants’ actions are at least objectively reckless as to the risk of infringement and this objective risk was either known or should have been known by Defendants.

281. Defendants’ direct and indirect infringement of the Asserted Patents is and has been willful, intentional, deliberate, and/or in conscious disregard of SoftView’s rights under the Asserted Patents.

282. SoftView has incurred substantial damages, including monetary damages.

283. SoftView has been irreparably harmed by Defendants’ infringement of the ’154 patent.

284. Therefore, SoftView is entitled to actual and/or compensatory damages, reasonable royalties, pre-judgment and post-judgment interest, enhanced damages, and costs.

COUNT IV – INFRINGEMENT OF U.S. PATENT NO. 8,533,628

285. SoftView incorporates by reference and re-alleges the foregoing paragraphs of this Complaint as if fully set forth herein.

286. Defendants have directly infringed at least claim 2 of the ’628 patent.

287. As a matter of law, the notice provisions of 35 U.S.C. § 287 do not apply in this case for the ’628 patent, and therefore SoftView has satisfied its obligations under 35 U.S.C. § 287 with respect to the ’628 patent. In addition, SoftView satisfied its obligations under 35 U.S.C. § 287 as to the ’628 patent at least by sending Defendants actual notice of infringement via letters dated August 15, 2022; October 23, 2023; and February 17, 2024. The letters made a specific charge of infringement of the ’628 patent by specific accused products, including a list of Android smartphones and tablets.

288. Defendants have directly infringed (literally and/or under the doctrine of equivalents) at least claim 2 of the ’628 patent in violation of 35 U.S.C. § 271 *et seq*., through use, including testing of the Accused products without authority or license of the Accused Products.

289. At the time of the claimed invention, reproducing web pages on a variety of devices presented a significant technical problem. Most internet content displayed as a flat single resolution with no browser support for zoom. ’353 Patent, 2:1-3. Much of the content on the internet was designed for display on desktop computers with a single target resolution (e.g., 640x480, 800x600). ’353 Patent, 2:4-5. Major internet content providers chose to create their Web pages using fixed resolution structures, such as tables, which gave them the ability to control the look and feel of their Web sites. ’353 Patent, 2:5-10. The fixed resolution approach was so

ubiquitous that it became the most common method to brand or uniquely identify websites. '353 Patent, 2:10-13.

290. While the fixed resolution approach was useful for site branding and product differentiation, it presented a daunting technical problem: how to display Internet content designed for desktop computers on devices with smaller screens, lower resolution, and/or different aspect ratios, such as cell phones and handheld computers. '353 Patent, 2:13-18. For example, a website designed for the fixed resolution of a larger desktop display would not fit correctly on small screens, low resolution screens, or devices with different aspect ratios.

291. The '628 patent claimed invention provides a solution to this daunting technical problem. Claims 1 and 2 recite as follows:

> 1. A method for enabling a user to perform full page browsing of a Web page on a hand-held wireless device in a manner that replicates browsing the Web page on a desktop Web browser, wherein the Web page comprises HTML-based content and cascading style sheet (CSS) content defining a page layout and design of display content on the Web page, the method comprising:
> in response to a user request to access the Web page, retrieving HTML-based content and the CSS content corresponding to the Web page and processing the HTML-based content and CSS content with the Web browser rendering engine to interpret the page layout and design of the display content on the Web page at a first scale; and
> employing scaled page layout information and/or content derived therefrom to generate scaled Web page content at multiple scale factors; and
> enabling a user of the mobile device to zoom in on an image via a user input to a touchscreen display and wherein said scaled Web page content including one or more images is generated by scaling bounding boxes corresponding to the one or more images to produce scaled bounding boxes and, for each of the one or more images, scaling content associated with the image to produce a scaled image that fits within its corresponding scaled bounding box;
> and enabling the user to perform full page browsing of the Web page at multiple zoom levels by generating corresponding views of the Web page on the hand-held device under which the page layout and design of the display content is retained relative to the interpretation of the page layout and design of the display content at the first scale,
> wherein each of the Web browser rendering engine and a Web browser rendering engine employed by a desktop browser for which browsing is replicated are the same.

> 2. The method of claim 1, wherein the views of the Web page are generated by employing a vector-based rendering engine to effect scaling of the display content as interpreted at the first scale.

292. As claimed in claim 1 of the ’628 patent, HTML-based content and CSS content corresponding to the web page is retrieved and processed by the Web browser rendering engine.

293. Scaled page layout information and/or content derived therefrom is employed to generate scaled Web page content at multiple scale factors. At the time of the claimed invention, this process was not well-known, routine, or conventional. Nor was it well-known, routine, or conventional to enable a user of a mobile device to zoom in on a Web page image via user input to a touchscreen display, nor utilize scaling bounding boxes corresponding to the one or more images to produce scaled bounding boxes, and for each of the one or more images, scaling content associated with the image to produce a scaled image that fits within its corresponding scaled bounding box. Nor was it well-known, routine, or conventional to enable the user to perform full page browsing of the Web page at multiple zoom levels by generating corresponding views of the Web page on the hand-held device under which the page layout and design of the display content is retained relative to the interpretation of the page layout and design of the display content at the first scale, nor wherein each of the Web browser rendering engine and a Web browser rendering engine employed by a desktop browser for which browsing is replicated are the same.

294. Claim 2 further provides that the views of the Web page are generated by employing a vector-based rendering engine to effect scaling of the display content as interpreted at the first scale. This was not well-known, routine, or conventional.

295. Claim 2 of the ’628 patent is directed to a technological solution to a technological problem, and was not well-known, routine, or conventional.

296. The advantages of the ’628 patent’s claimed technical solution are substantial. Web pages could now “be rapidly rendered, zoomed, and panned. Moreover, the rendered displays provide substantially the same or identical layout as the original Web page, enabling users to easily navigate to selected content and features on familiar Web pages.” ’353 Patent, Abstract. This meant that users could finally be presented with web pages that fit and/or were effectively usable on devices with smaller screens, different resolutions, or different aspect ratios.

297. With the claimed invention, the web page could be presented as it would be on a desktop computer (as most web pages were designed for), and then rapidly panned and zoomed to facilitate use on devices with smaller screens, different resolutions, or different aspect ratios. A Web page that was unusable when presented with the fixed resolution approach could now be zoomed to suit the device display. Today, this functionality is pervasive on smartphone and tablet devices and taken for granted. But prior to the claimed invention, it did not exist.

298. In addition, the claimed invention allowed the look and feel of the Web page to be maintained, even when designed for fixed resolutions. This allowed content providers to continue branding their Web pages with fixed resolution designs. It also allowed users to access Web pages as they were used to seeing them on desktop computers suited to their fixed resolution design, which meant users knew where to locate items on the page based on its familiar look and feel. Thus, the claimed invention made web pages more convenient for users.

299. The claimed invention has an additional advantage: internet content providers can rely on the fixed resolution approach instead of designing numerous different versions of the same web page to suit the screen size, resolution, and aspect ratio of different devices. For example, a web page designed for a 16-inch desktop display could be effectively viewed and used on a small hand-held mobile phone. This is superior to the alternative approach of designing multiple

versions of the same web page to suit different devices, which was time consuming, costly, and impractical for many Web content developers.

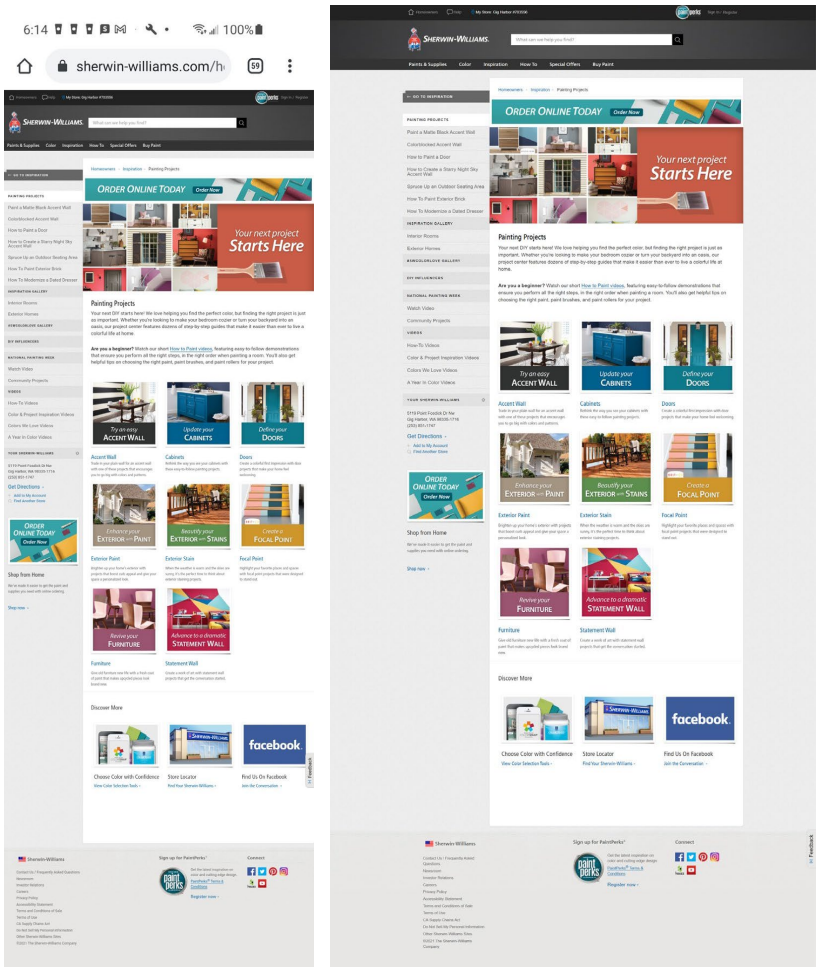
300. Claim 1 of the ’628 patent recites, “A method for enabling a user to perform full page browsing of a Web page on a hand-held wireless device in a manner that replicates browsing the Web page on a desktop Web browser, wherein the Web page comprises HTML-based content and cascading style sheet (CSS) content defining a page layout and design of display content on the Web page.” To the extent the preamble is limiting, the Accused Products satisfy it.

301. The Accused Products, such as the Samsung Galaxy S10, are hand-held wireless devices that enable users to perform full page browsing of a Web page, such as in the Samsung Internet and Chrome browsers, in a manner that replicates browsing the Web page on a desktop Web browser.



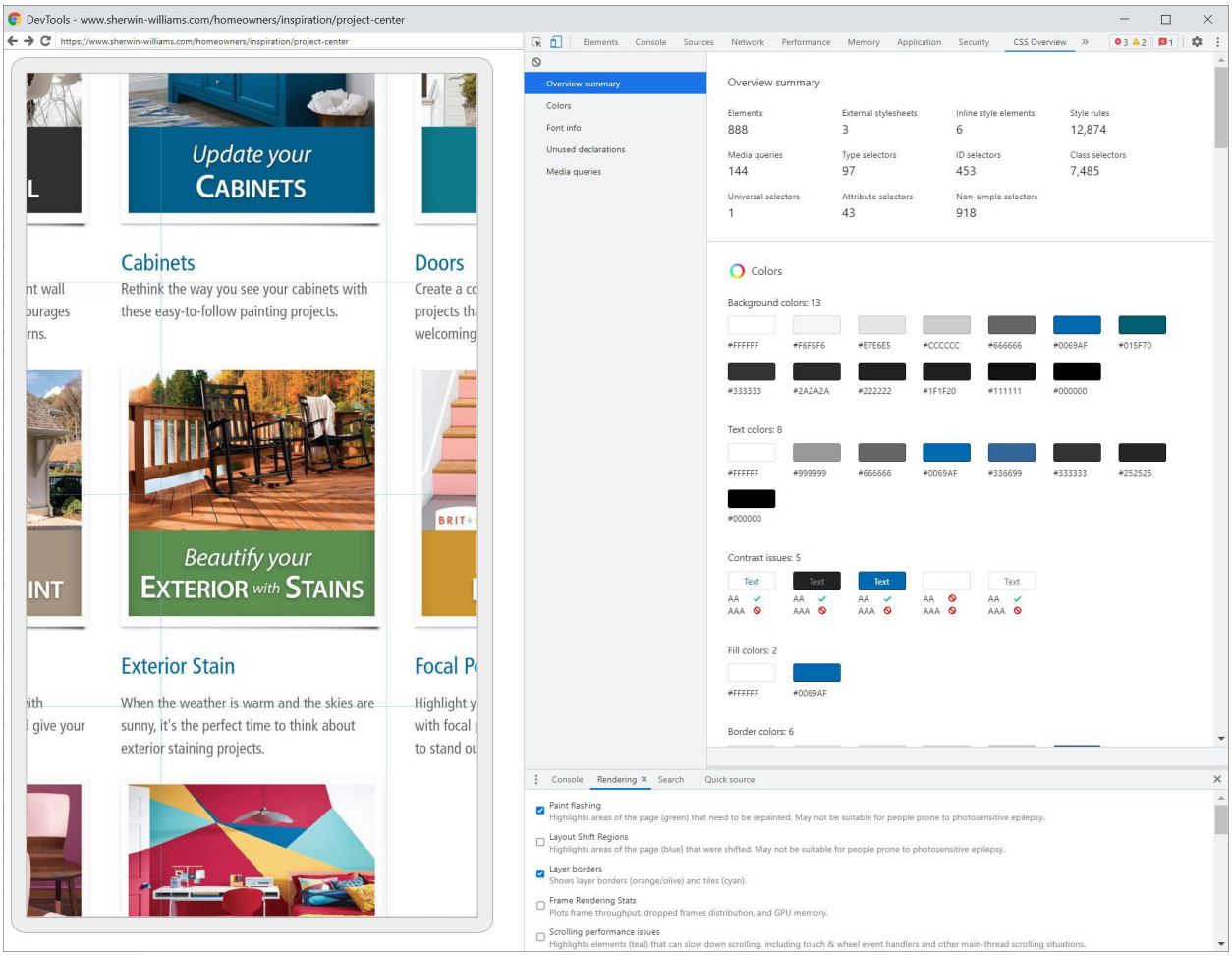
https://www.gsmarena.com/samsung_galaxy_s10-pictures-9536.php#.

302. The screenshot below on the left was taken on a Web page using the Chrome browser in the Galaxy S10. The screenshot below on the right was taken using the Chrome desktop browser on a MacBook Pro running Windows 10 Pro.



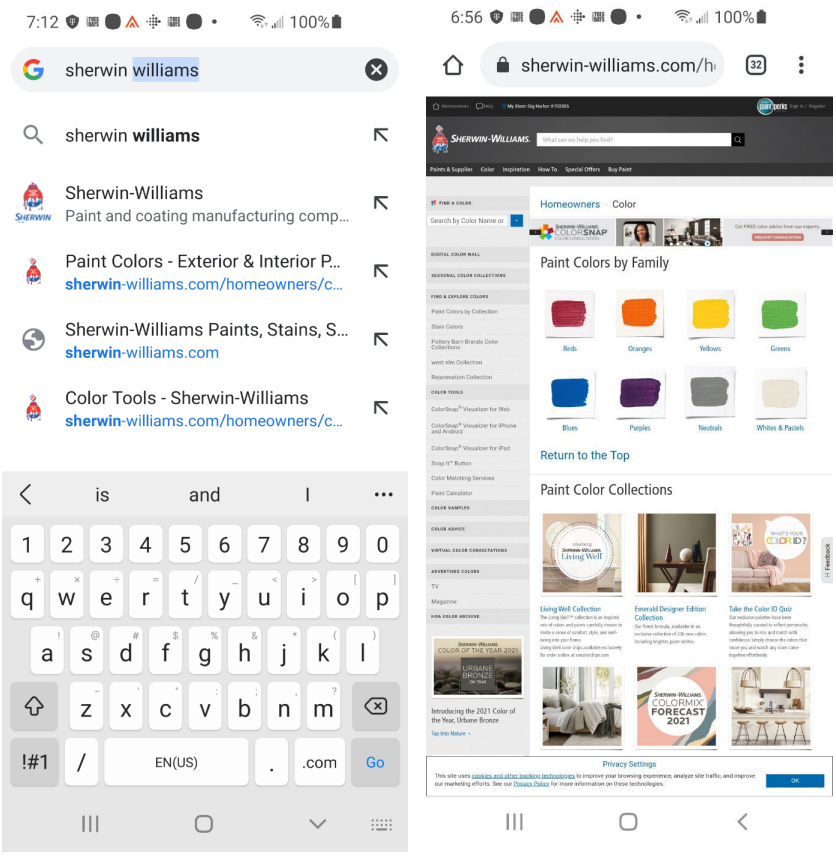
303. The Web page comprises HTML-based content and CSS content defining a page layout and design of display content on the Web page.

304. Below is a CSS Overview for the above Web page in the Chrome Web Inspector.



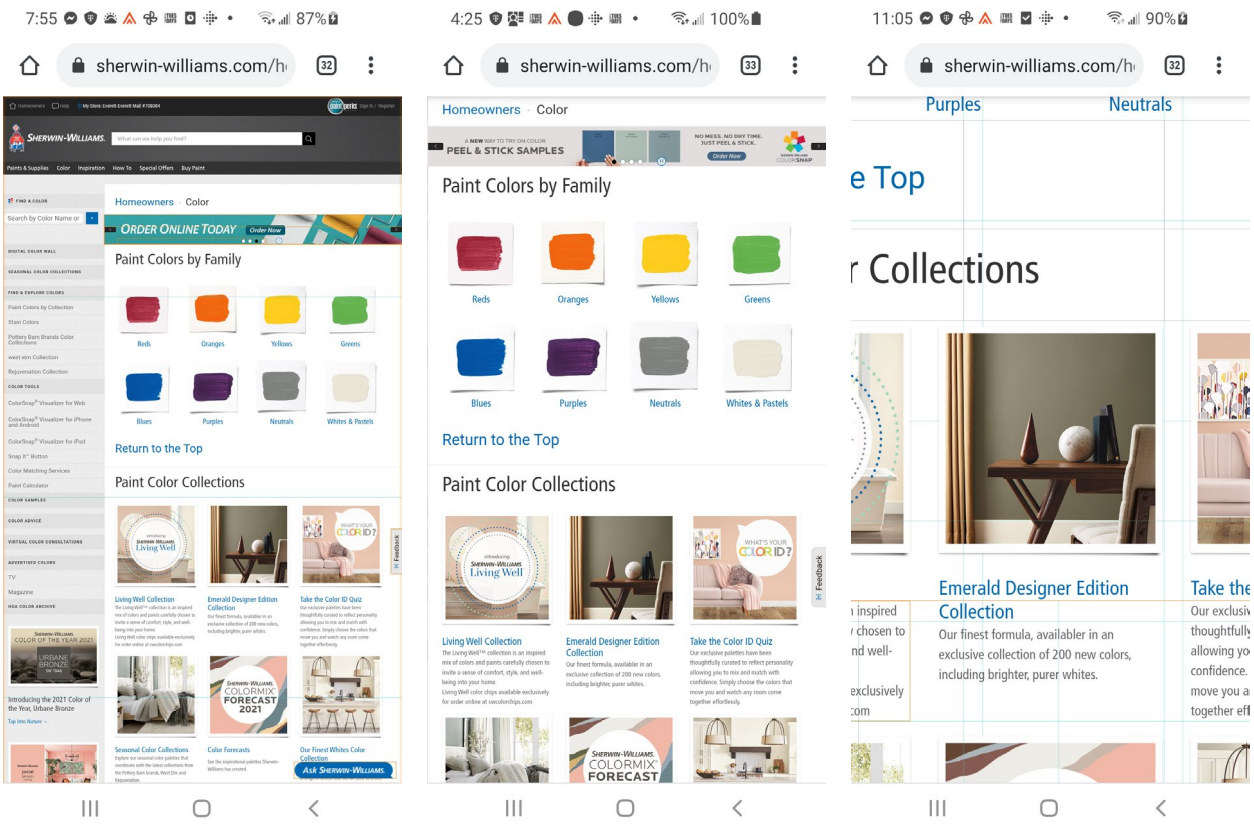
305. Claim 1 of the ’628 patent recites “in response to a user request to access the Web page, retrieving HTML-based content and the CSS content corresponding to the Web page and processing the HTML-based content and CSS content with the Web browser rendering engine to interpret the page layout and design of the display content on the Web page at a first scale.” The Accused Products perform these operations.

306. The screenshots below show that a user can request access to a Web page and in response, HTML-based content and the CSS content corresponding to the Web page is retrieved. The screenshots further show processing the HTML-based content and CSS content with the Web browser rendering engine to interpret the page layout and design of the display content on the Web page at a first scale.



307. Claim 1 of the ’628 patent recites “employing scaled page layout information and/or content derived therefrom to generate scaled Web page content at multiple scale factors.” The Accused Products perform this operation.

308. The below screenshots taken on the Galaxy S10 in the Chrome browser reflect employing scaled page layout information and/or content derived therefrom to generate scaled Web page content at multiple scale factors.



309. Blink processes HTML to produce a document object model (DOM) tree and employs CSS rules and the CSS box model to generate a Layout Tree with page layout. Many of the core rendering processes employed by WebKit and Blink are similar as Blink and WebKit share the same core rendering code (WebCore). Blink uses a separate CSS layout engine called LayoutNG and uses a separate JavaScript engine. Aspects of the main flow for Blink are similar to the main flow for WebKit below, including use of HTML and CSS parsers and generation of a Document Object Model (DOM) and DOM Tree.

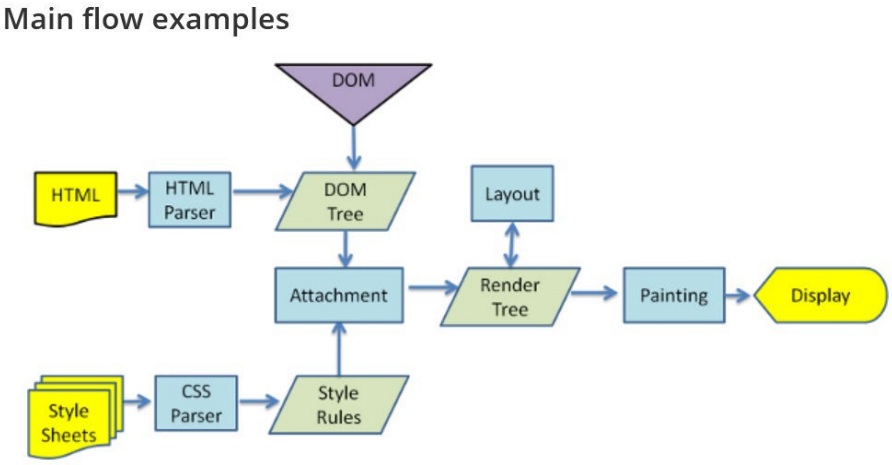


Figure : WebKit main flow

https://www.html5rocks.com/en/tutorials/internals/howbrowserswork/.

310. LayoutNG generates a Layout tree with bounding boxes (called boxes for short) employing the CSS box model.

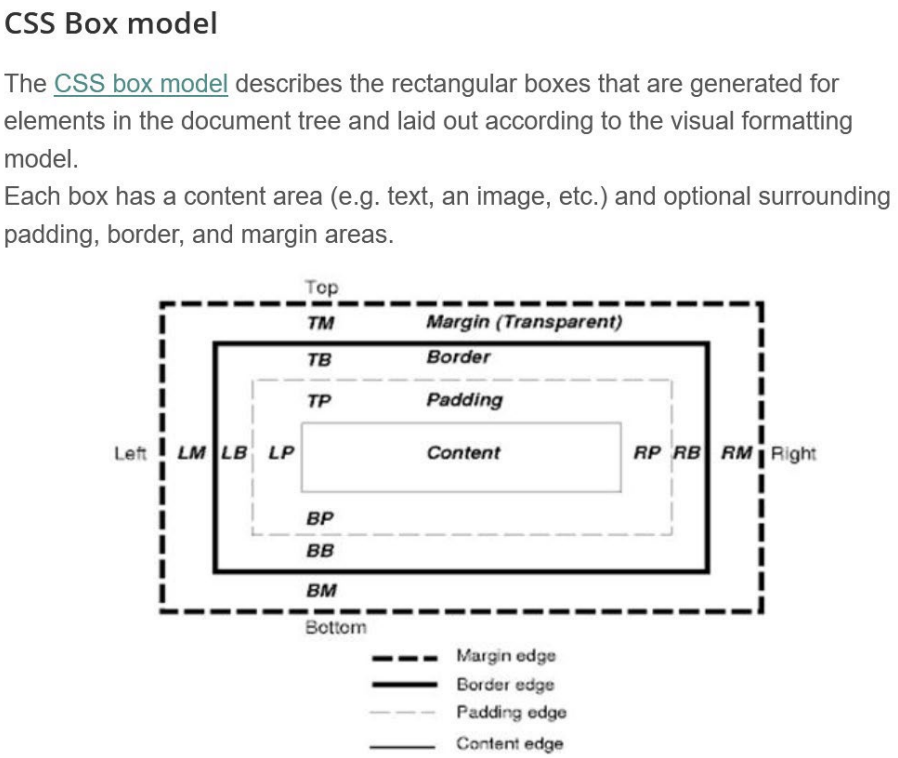


Figure : CSS2 box model

https://www.html5rocks.com/en/tutorials/internals/howbrowserswork/.

311. LayoutNG uses Logical & Physical Units.

Logical & Physical Units

To aid in readability and to ensure proper conversion between coordinate systems a set of dedicated units will be used to represent offsets, locations, sizes, and rects for each coordinate system where they are applicable. Conversion between coordinate systems must be done explicitly and no implicit conversion will be allowed.

Logical Units

NGLogicalOffset and NGLogicalSize represents an offset or a size. Each has a pair of LayoutUnits [11] that

represents the offset or size in the inline and block directions.

NGLogicalRect represents a rectangle and contains an offset and a size, represented by the two units described above.

These are all in a logical coordinate space in the sense that they do not take writing mode or directionality into account.

Physical Units

NGPhysicalOffset, NGPhysicalLocation, and NGPhysicalSize represents an offset, location or a size. The location unit is different from the offset one in that it represents the resolved location from the top of the tree while the offset unit represents the delta from the parent. Each has a pair of LayoutUnits that represents the offset or size in the horizontal and vertical directions.

NGPhysicalRect represents a rectangle and contains an offset and a size, represented by the two units described above.

These are all in a physical coordinate space in the sense that they represent coordinates in a top-to-bottom, left-to-right coordinate system. Converting from a logical unit to a physical one thus requires the coordinates to be resolved. The *Writing Modes and Coordinate Systems* section goes into more details.

https://docs.google.com/document/d/1uxbDh4uONFQOiGuiumlJBLGgO4KDWB8ZEkp7Rd47fw4/edit?tab=t.0#heading=h.guvbepjyp0oj.

312. LayoutNG generates a Layout Tree comprising NGFragment and NGText Nodes.

Layout Tree

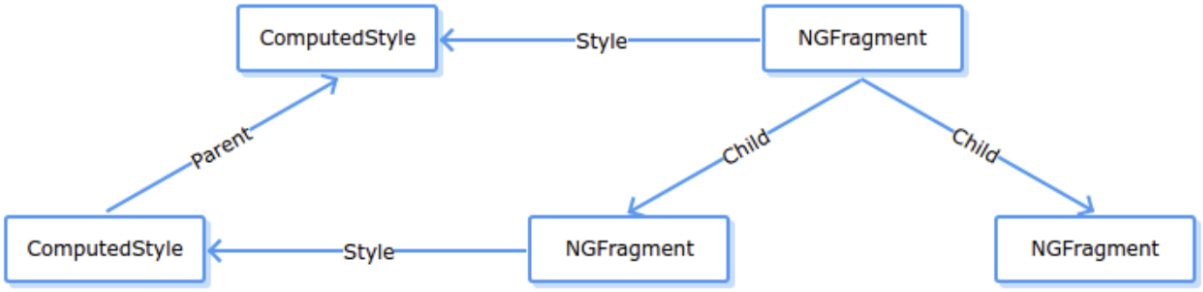


Figure shows the style tree (left) and layout tree (right) and their relationship.

The layout tree is made up entirely of NGFragment and NGText nodes and is immutable. Each node includes a back pointer to the computed style, a list of children and the size and location of the node itself. The location is relative to the immediate parent.

Given the immutability of the tree, use of relative coordinates, and the lack of a parent pointer entire subtrees may be reused across layouts. This concept is discussed in detail in the subsequent layouts section.

https://developer.chrome.com/docs/chromium/blinkng.

313. The Layout Tree also represents the box hierarchy of the HTML document.

Pipelining style, layout, and pre-paint

Collectively, the rendering phases before *paint* are responsible for the following:

- Running the *style cascade* algorithm to calculate final style properties for DOM nodes.
- Generating the layout tree representing the box hierarchy of the document.
- Determining size and position information for all boxes.
- Rounding or snapping sub-pixel geometry to whole pixel boundaries for painting.
- Determining the properties of composited layers (affine transformation, filters, opacity, or anything else that can be GPU accelerated).
- Determining what content has changed since the previous paint phase, and needs to be painted or repainted (paint invalidation).

https://developer.chrome.com/docs/chromium/blinkng.

314. LayoutNG uses ClientRects to size and position HTML Elements (DOM objects) relative to the viewport.

ClientRect	Size and position of an Element relative to the viewport.

Walking up the Tree

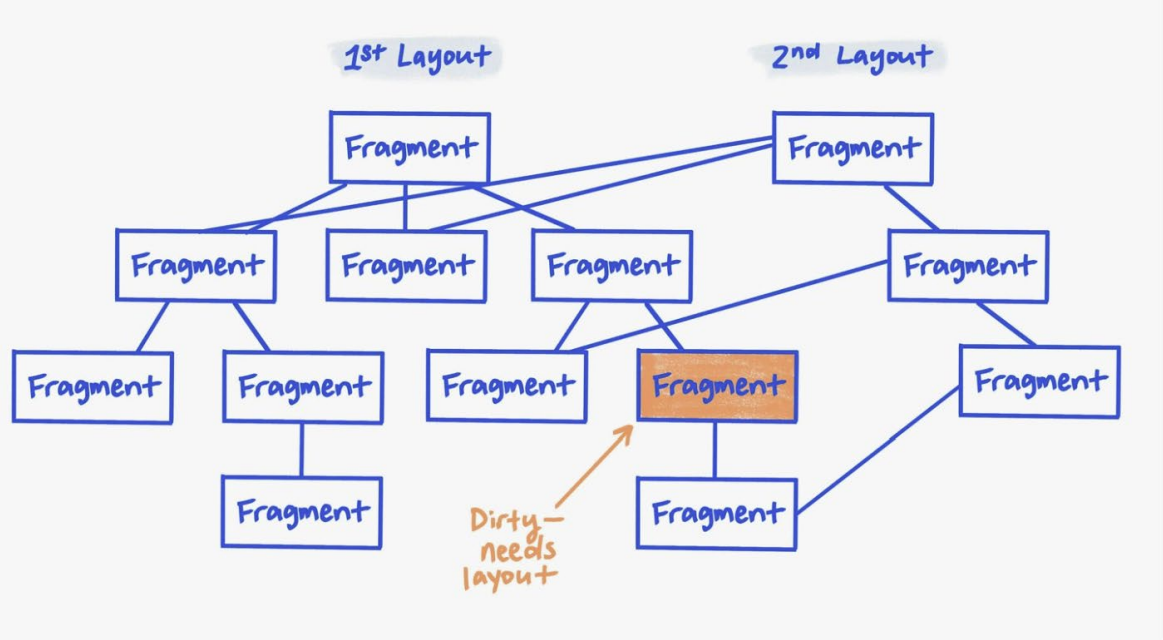
ClientRect calculations and IntersectionObserver need to perform a tree-walk up to the root in order to apply transform/layout information. Transform information is not on the layout tree. Adding layout-parent pointers in the Layout Tree would mean that we lose the immutability property; so instead the parent-layout pointers will be stored on the DOM tree. These operations will begin at the DOM node, ask its NGUnplacedFragment for layout information, then walk up the DOM tree to its parent-layout and so forth.

https://docs.google.com/document/d/1uxbDh4uONFQOiGuiumlJBLGgO4KDWB8ZEkp7Rd47fw4/edit?tab=t.0#heading=h.guvbepjyp0oj.

315. Blink generates an immutable fragment tree, which is used to represent the position and size of all elements on the page (without transforms applied).

The immutable fragment tree

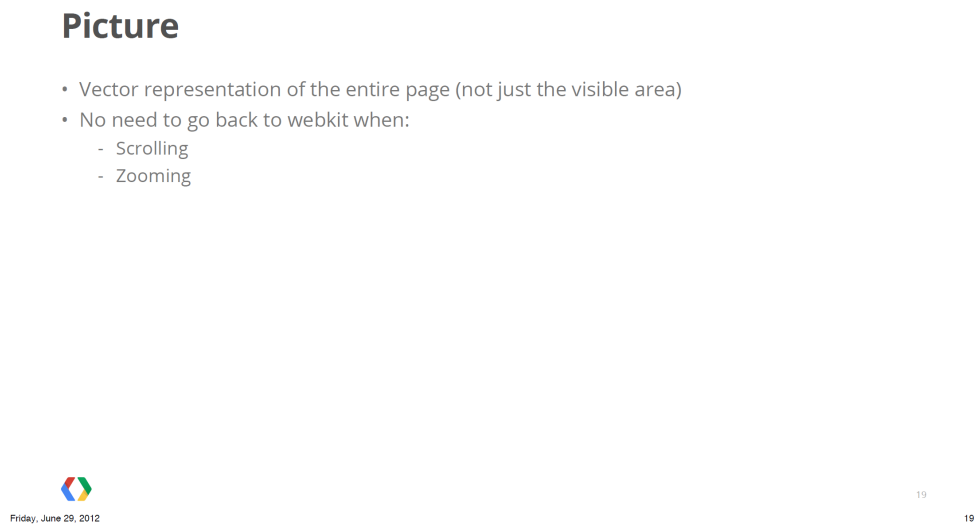
The immutable fragment tree is the output of the layout stage of the rendering pipeline. It represents the position and size of all elements on the page (without transforms applied).



Each fragment represents a part of a DOM element. Typically there is only one fragment per element, but there can be more if it is split across different pages when printing, or columns when in a multi-column context.

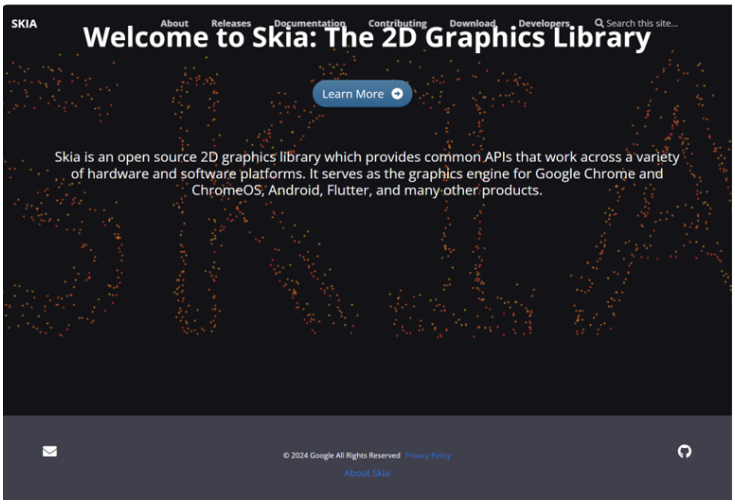
https://developer.chrome.com/docs/chromium/renderingng-data-structures.

316. WebView (and the Samsung Internet and Chrome browsers) creates a vector representation of the entire page, which is scalable.



Android WebView — Google IO 2012.

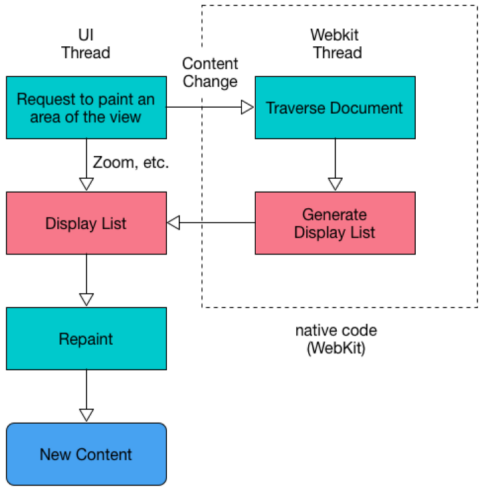
317. The Samsung Internet and Chrome browsers use the Skia 2D vector graphics rendering software/library, which is also referred to as the Skia Graphics Engine.



318. WebView (and the Samsung Internet and Chrome browsers) records lists of painting instructions for the full page and builds display lists. Zooming is accomplished by changing scale and translation parameters in a transformation matrix (SkMatrix) to scale and/or translate position vector coordinates and calling the painting instructions in the display list. While the following describes the approach used by the Chrome browser with WebKit, the current

painting process using Blink is similar. The display list model functions as a vector representation of the Web page, including for each object on the page such as text and images.

> So, instead of directly asking webkit to *paint* the screen, the team had a better idea. They decided to modify Webkit to *record* the list of painting instructions for the full page, but to not paint directly — basically using a "display list" model. The list of painting commands were for example "draw this text here", or "draw this image there".

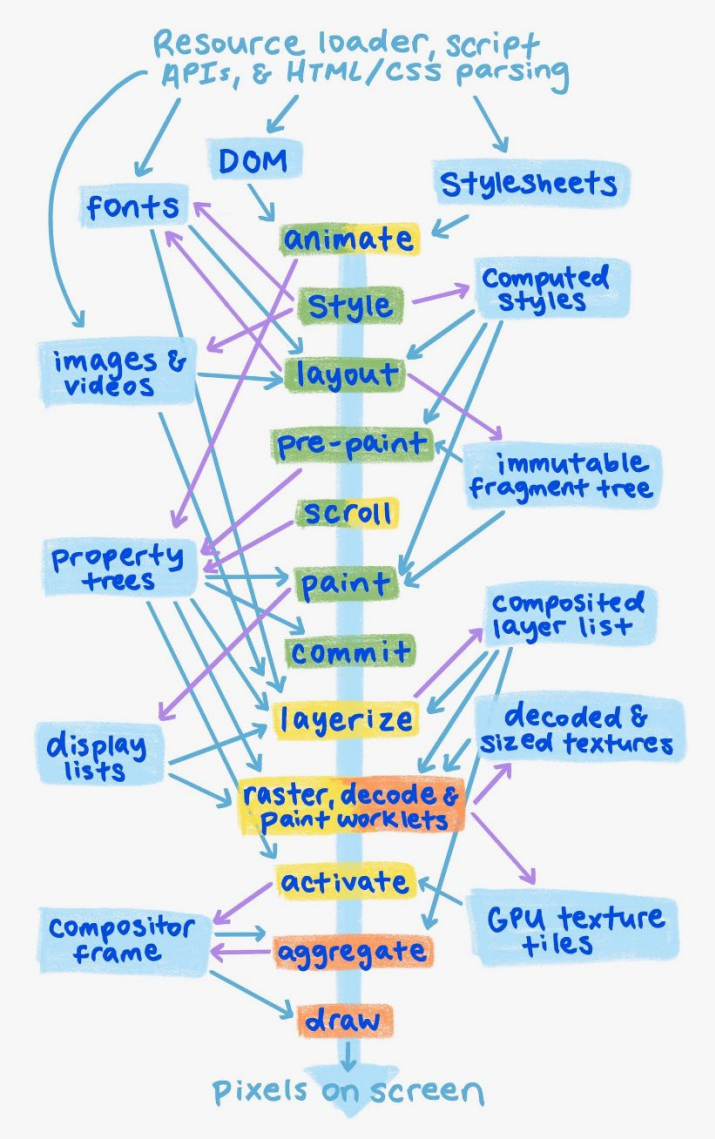


Rendering loop (Android)

> When the user would need to zoom in for example, it was not necessary to ask Webkit to repaint the screen — the webview instead could just replay the already present display list, only changing the zoom factor. This also gave a rather nice side-effect that the screen was always sharp — no text blurriness when zooming in, no missing content, as the content was always being redrawn "the right way".
>
> You could think of this display list as a way to have a vector representation of the entire page, that you could scale on demand and redraw — the difference between illustrator and photoshop.

https://medium.com/@camaelon/what-s-in-a-web-browser-83793b51df6c.

319. Display lists are used in the current Rendering Pipeline.


Resource loader, script APIs, & HTML/CSS parsing
DOM
Stylesheets
fonts
animate
Computed styles
Style
images & videos
layout
pre-paint
immutable fragment tree
scroll
property trees
paint
composited layer list
commit
layerize
decoded & sized textures
display lists
raster, decode & paint worklets
activate
GPU texture tiles
compositor frame
aggregate
draw
Pixels on screen

The stages are:

1. *Animate:* change computed styles and mutate *property trees* over time based on declarative timelines.
2. *Style:* apply CSS to the DOM, and create *computed styles*.
3. *Layout:* determine the size and position of DOM elements on the screen, and create the *immutable fragment tree*.
4. *Pre-paint:* compute property trees and invalidate any existing *display lists* and GPU *texture tiles* as appropriate.
5. *Scroll:* update the scroll offset of documents and scrollable DOM elements, by mutating property trees.
6. *Paint:* compute a display list that describes how to raster GPU texture tiles from the DOM.
7. *Commit:* copy property trees and the display list to the compositor thread.
8. *Layerize:* break up the display list into a *composited layer list* for independent rasterization and animation.
9. *Raster, decode and paint worklets:* turn display lists, encoded images, and paint worklet code, respectively, into GPU texture tiles.
10. *Activate:* create a *compositor frame* representing how to draw and position GPU tiles to the screen, together with any visual effects.
11. *Aggregate:* combine compositor frames from all the visible compositor frames into a single, global compositor frame.
12. *Draw:* execute the aggregated compositor frame on the GPU to create pixels on-screen.

https://developer.chrome.com/docs/chromium/renderingng-architecture#pipeline.

Display lists and paint chunks

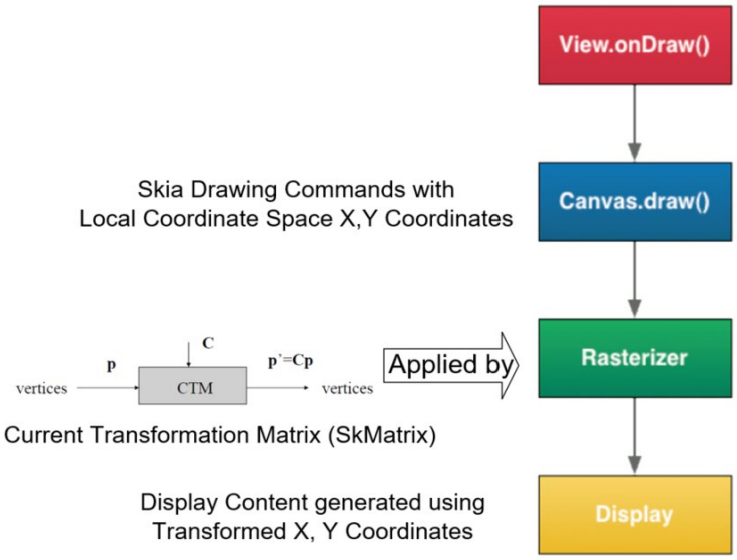
A display item contains low-level drawing commands (see here) that can be rasterized with Skia. Display items are typically simple, with just a few drawing commands, such as drawing a border or background. The paint tree walk iterates over the layout tree and associated fragments following CSS painting order to produce a display item list.

For example:



https://developer.chrome.com/docs/chromium/renderingng-data-structures#display_lists_and_paint_chunks.

320. Skia uses a Current Transformation Matrix (SkMatrix) for scaling and translation of content. An example is shown below for Skia painting.



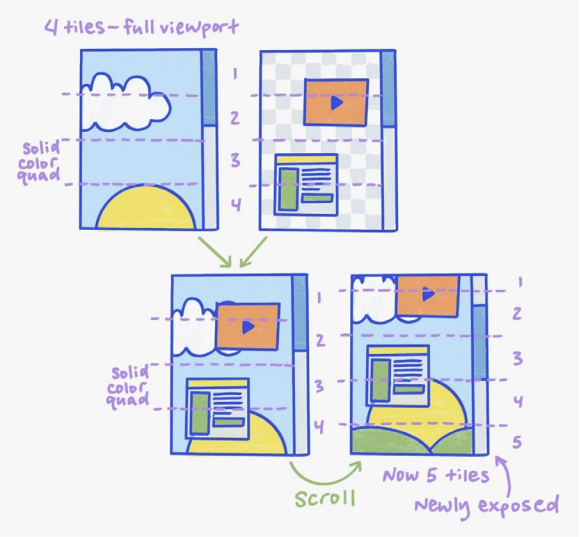
Android Accelerated Rendering Google IO 2011 (modified).

321. The rendering pipeline uses the Skia display lists to rasterize pixels onto GPU texture tiles.

Tiles

In theory, a render process or browser process compositor could rasterize pixels into a single texture the full size of the renderer viewport and submit that texture to Viz. To display it, the display compositor would just have to copy the pixels from that single texture into the appropriate position in the frame buffer (for example, the screen). However, if that compositor wanted to update even a single pixel it would need to re-rasterize the full viewport and submit a new texture to Viz.

Instead, the viewport is divided into tiles. A separate GPU texture tile backs each tile with the rasterized pixels for part of the viewport. The renderer can then update individual tiles or even just change the position on screen for the existing tiles. For example, when scrolling a website, the position of existing tiles would shift up and only occasionally a new tile would need to be rasterized for content further down the page.

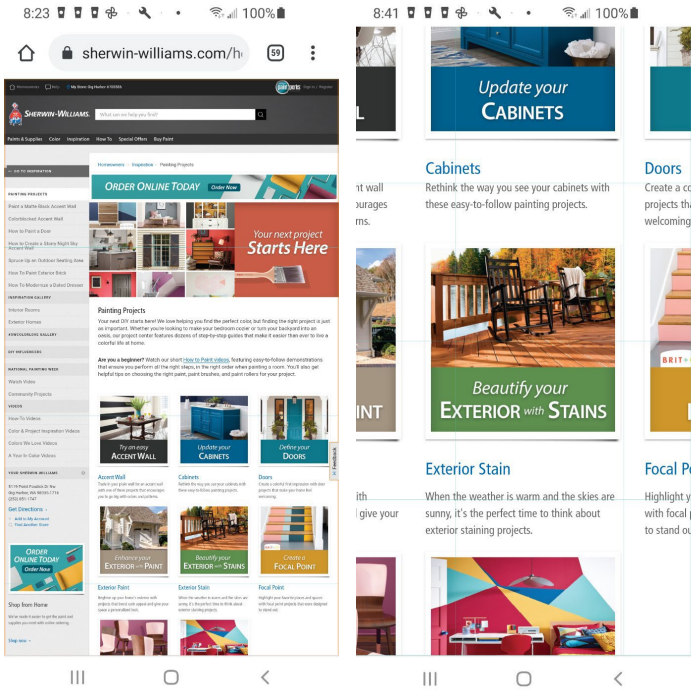


This image depicts an image of a sunny day, with four tiles. When a scroll occurs, a fifth tile begins to appear. One of the tiles happens to only have one color (sky blue), and there is a video and an iframe on top.

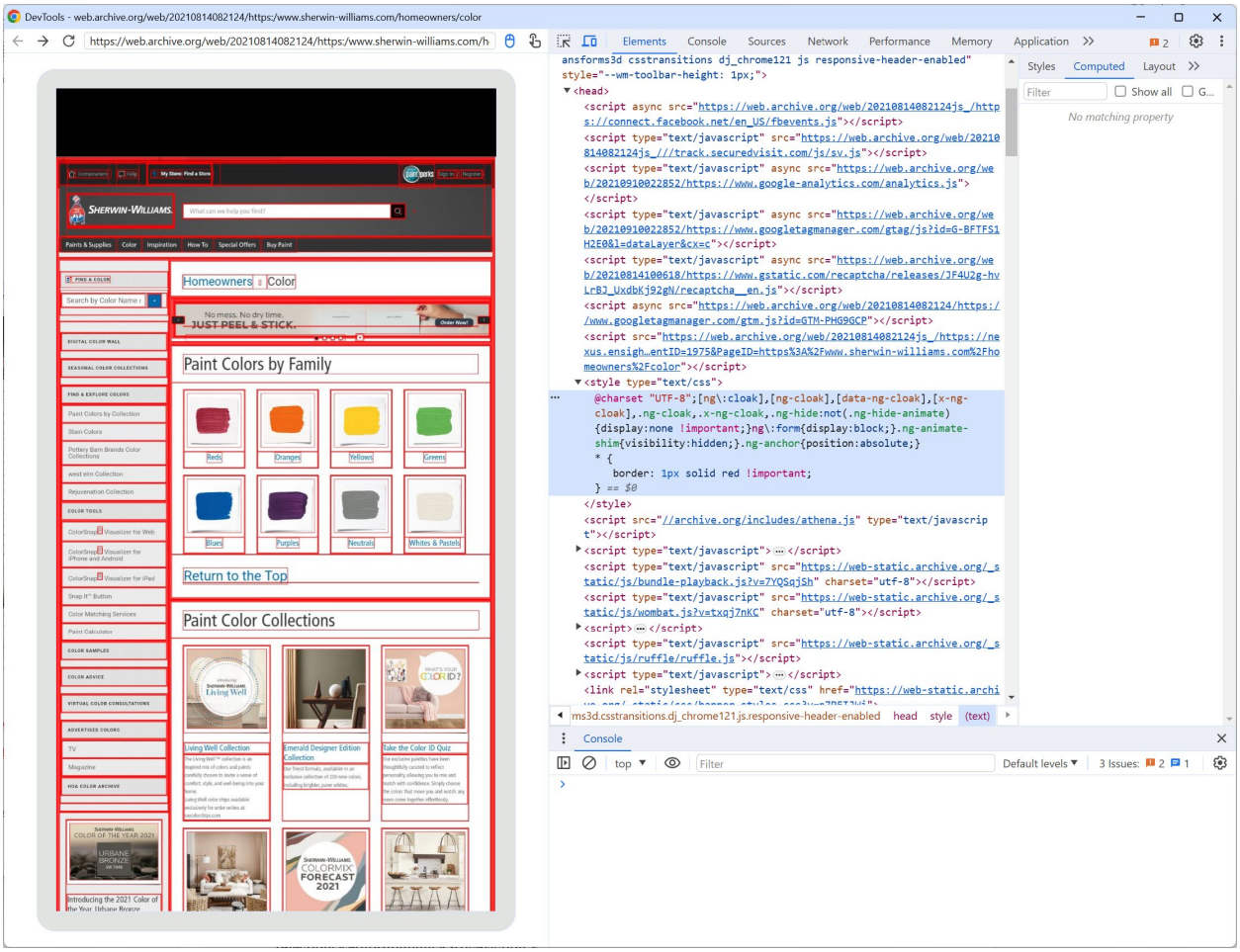
https://developer.chrome.com/docs/chromium/renderingng-data-structures#tiles.

322. Claim 1 of the ’628 patent recites “enabling a user of the mobile device to zoom in on an image via a user input to a touchscreen display and wherein said scaled Web page content including one or more images is generated by scaling bounding boxes corresponding to the one or more images to produce scaled bounding boxes and, for each of the one or more images, scaling content associated with the image to produce a scaled image that fits within its corresponding scaled bounding box.” The Accused Products perform these operations.

323. The screenshots below were taken of a Galaxy S10 running the Chrome browser. A user can zoom in on an image via user input to a touchscreen display, such as pinch zooming or double tapping on the image. This is done by scaling the image to produce a scaled image that fits within its corresponding scaled bounding box.



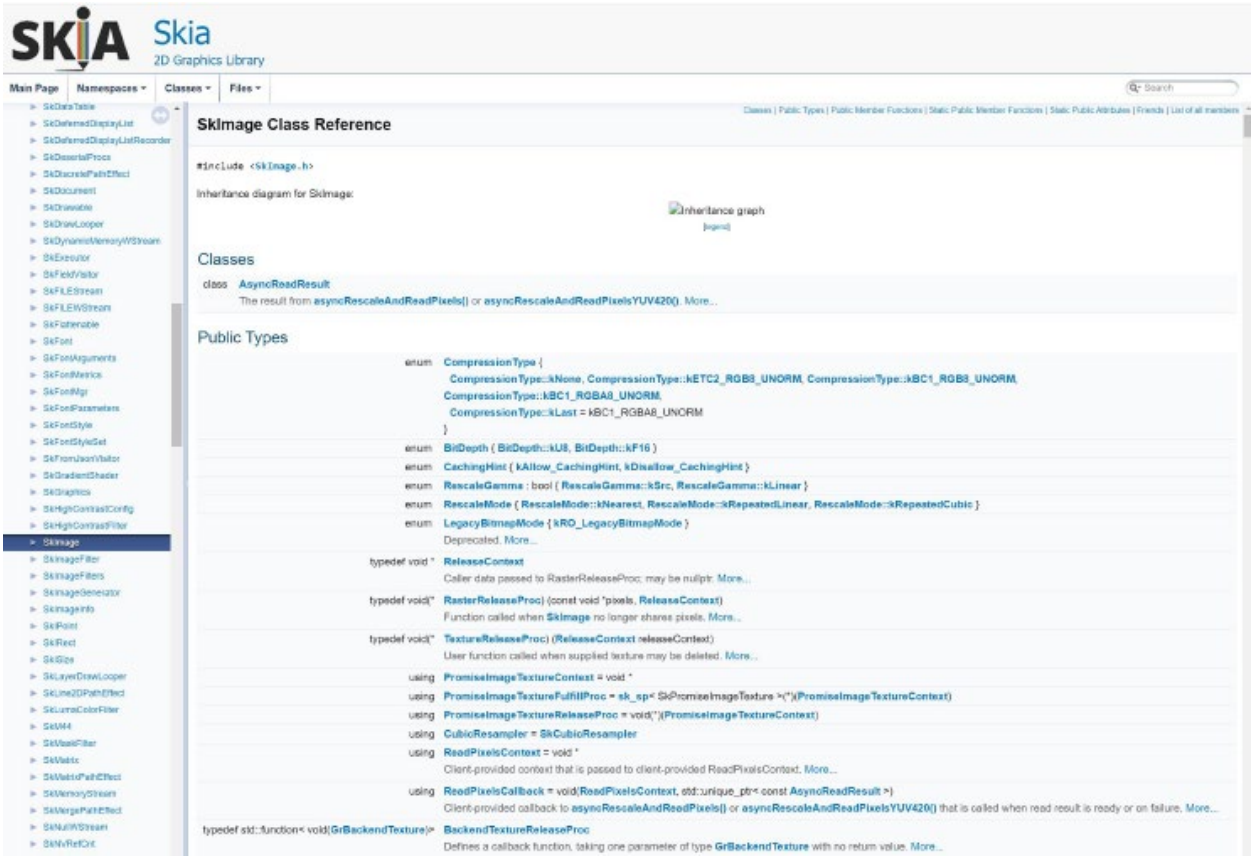
324. A visual representation of the Layout Tree structure can be shown by modifying the HTML document to render a visual border for each bounding box.



The following code is added to the end of the <style> section highlighted above:

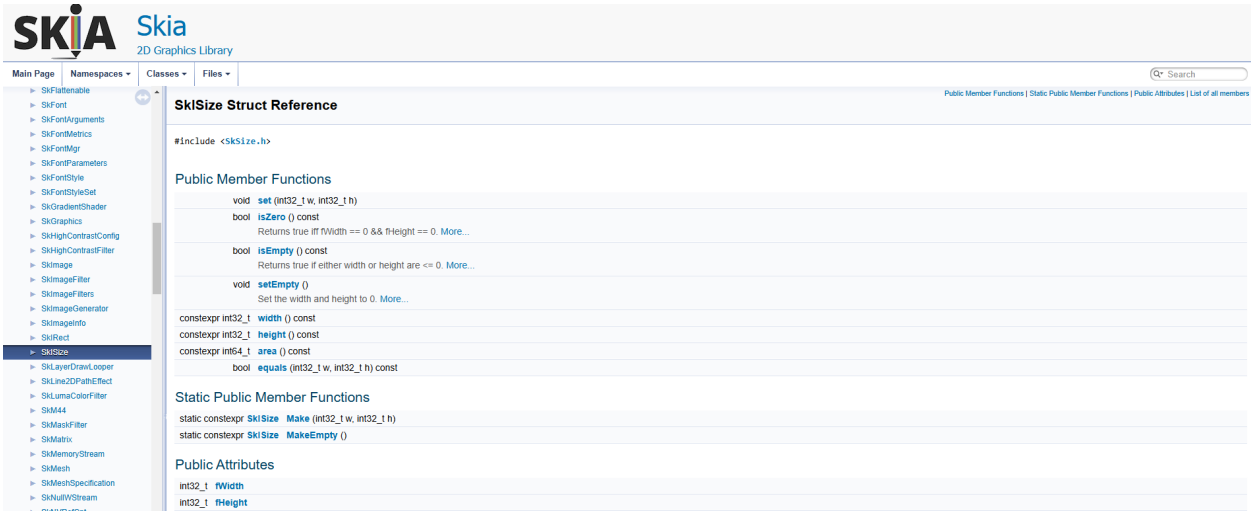
```
* {
  border: 1px solid red !important;
}
```

325. Skia utilized by the Samsung Internet and Chrome browsers includes an SkImage class for rendering images.



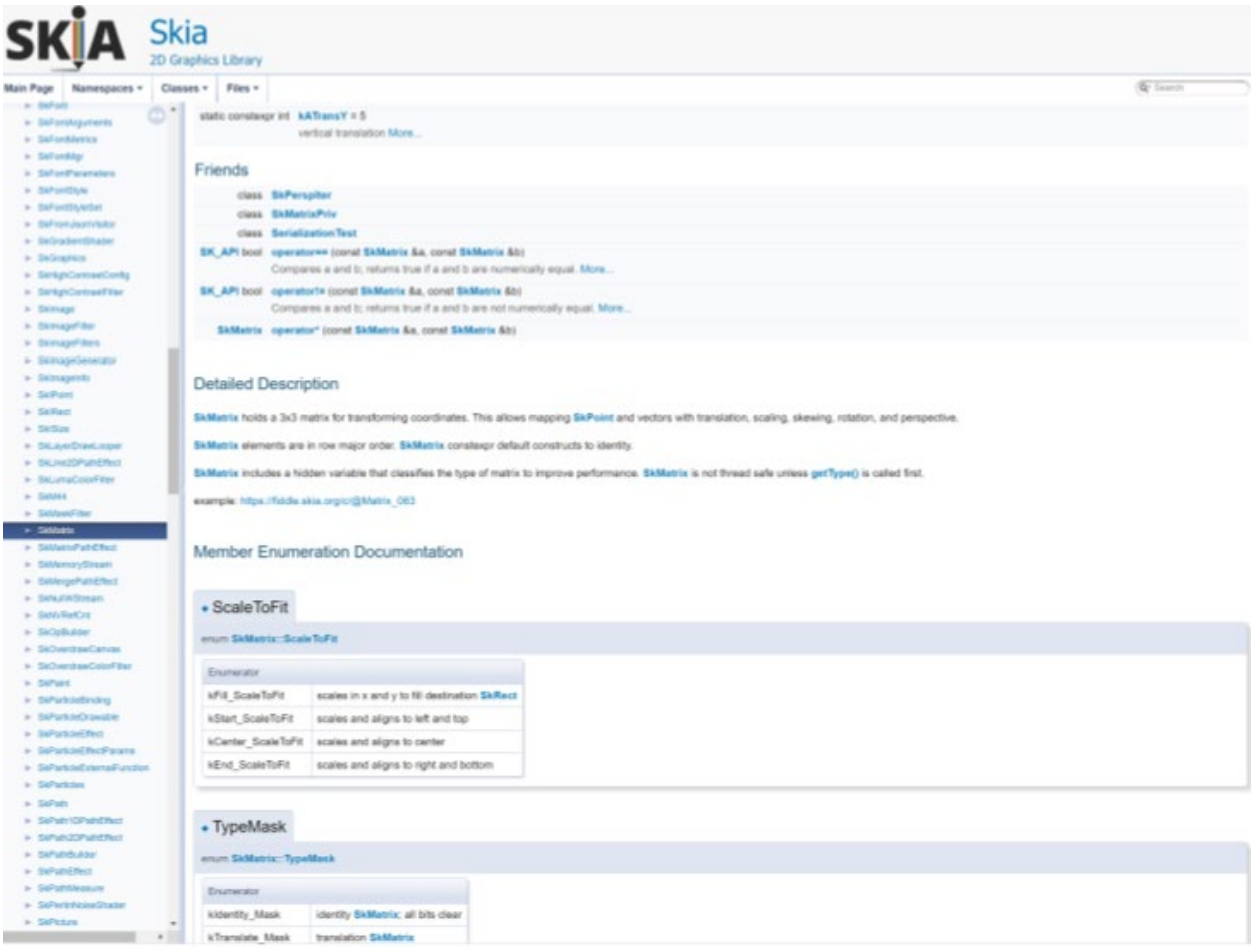
https://api.skia.org/classSkImage.html.

326. The SkImage class includes Public Member Functions that enable an image to be scaled by specifying the SkISize, which has a height and width. The original page content is laid out at full scale using the X,Y coordinates and dimensions generated by WebKit/Blink. Those include a SkISize structure for each image having a height and width of the DOMRect for the image in the DOM tree. The drawing (rendering) commands are added to a Display List and scaling is performed by re-rendering the drawing commands using a new transformation matrix.



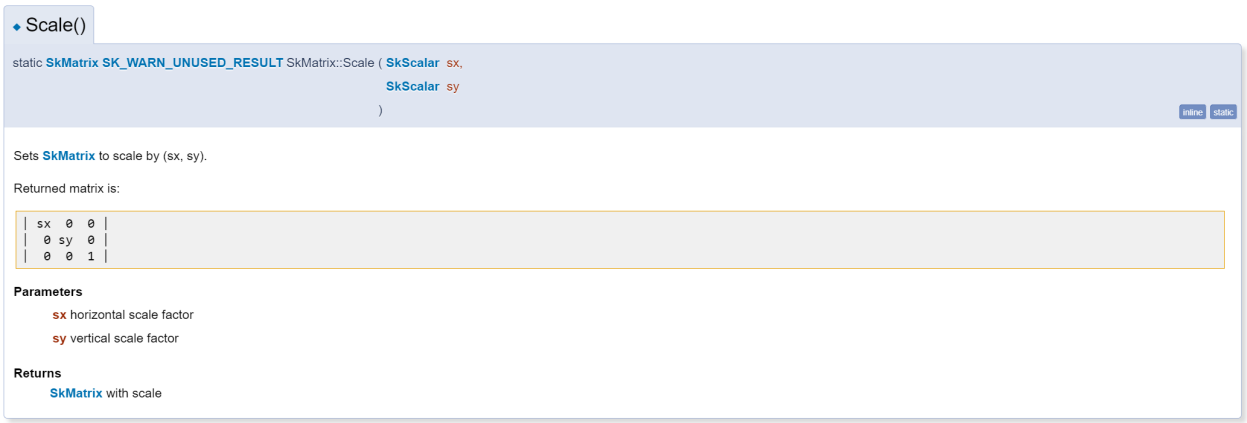
https://api.skia.org/structSkISize.html.

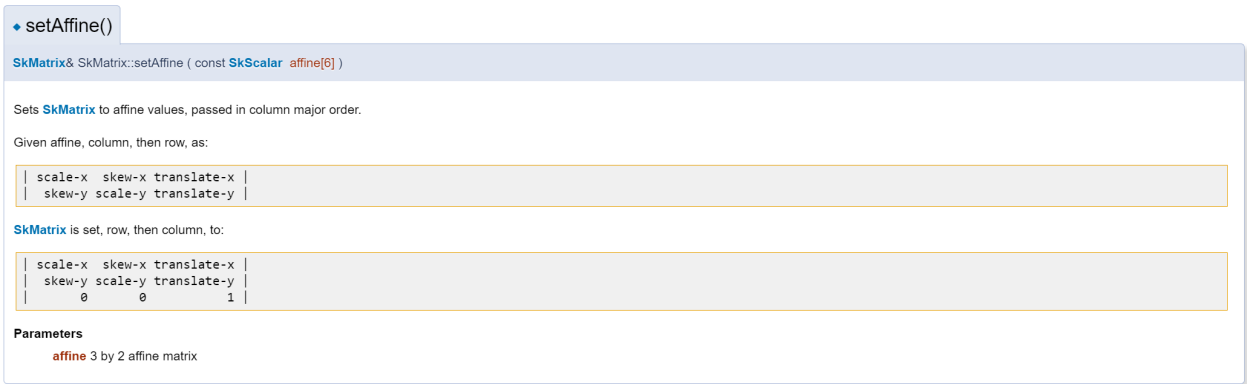
327. The Display List uses the SKMatrix class to effect scaling. SKMatrix holds a 3x3 matrix for transforming coordinates (also known as an affine transformation). SKMatrix is used to effect translation and scaling. Skia renders images by ScaleToFit the destination rectangle (SKRect). SKRect is scaled using the current SKMatrix, and then the Image is scaled to fit the SKRect that is generated.



https://api.skia.org/classSkMatrix.html.

328. The Skia Scale() function can be used to change the sx (horizontal) and sy (vertical) scale factors in the SKMatrix. The setAffine() function can be used to set values for the first two rows of the SKMatrix. This enables scaling and translation to be performed together.





◆ setAffine()

SkMatrix& SkMatrix::setAffine (const SkScalar affine[6])

Sets SkMatrix to affine values, passed in column major order.

Given affine, column, then row, as:

```
| scale-x  skew-x translate-x |
|  skew-y scale-y translate-y |
```

SkMatrix is set, row, then column, to:

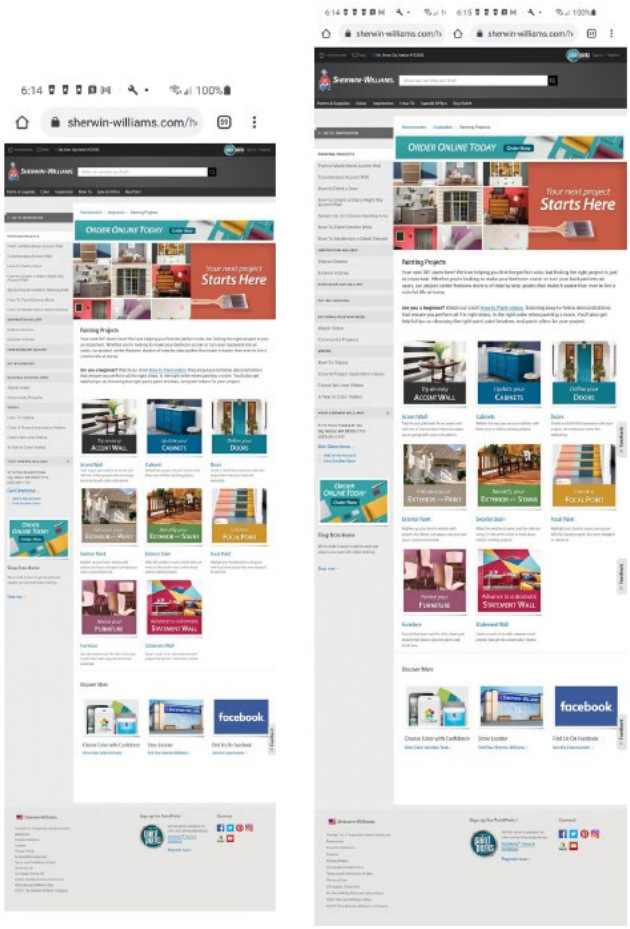
```
| scale-x  skew-x translate-x |
|  skew-y scale-y translate-y |
|       0       0           1 |
```

Parameters

affine 3 by 2 affine matrix

329. Claim 1 of the ’628 patent recites “enabling the user to perform full page browsing of the Web page at multiple zoom levels by generating corresponding views of the Web page on the hand-held device under which the page layout and design of the display content is retained relative to the interpretation of the page layout and design of the display content at the first scale.” The Accused Products perform this operation.

330. The screenshots below show the Chrome browser running on a Galaxy S10 at different zoom levels. The page layout and design of the display content is retained relative to the interpretation of the page layout and design of the display content at the first scale.



331. Claim 1 of the ’628 patent recites “wherein each of the Web browser rendering engine and a Web browser rendering engine employed by a desktop browser for which browsing is replicated are the same.” The Accused Products satisfy this requirement.

332. Blink is an HTML- and CSS-compliant rendering engine used by the Chrome desktop and mobile browsers, as well as the Samsung Internet browser.

333. Claim 2 of the ’628 patent recites, “wherein the views of the Web page are generated by employing a vector-based rendering engine to effect scaling of the display content as interpreted at the first scale.” The Accused Products satisfy this requirement.

334. As alleged above, the Accused Products employ Skia to effect scaling of the display content as interpreted at the first scale.

Samsung's Inducement of Infringement of the '628 Patent

335. SoftView incorporates by reference and re-alleges the foregoing paragraphs of this Complaint as if fully set forth herein.

336. The '628 patent was filed on April 15, 2007 and issued on September 10, 2013. On information and belief, Defendants had knowledge of the '628 patent on or soon after its issuance date.

337. Defendants were served on October 5, 2011, with the Second Amended Complaint in the Delaware litigation alleging infringement of the '353. The '628 patent is in the same patent family, was pending, and issued during the Delaware litigation.

338. On information and belief, Defendants had knowledge of the '628 patent or should have known of the '628 patent in consideration that Defendants were in prior litigation involving the '353 and '926 patents which are members of the same patent family.

339. SoftView sent Defendants notice letters dated August 15, 2022; October 23, 2023; and February 17, 2024.

340. Each letter notified Defendants of their infringement of the '628 patent.

341. Defendants did not respond to any of the three notice letters.

342. SoftView is informed and believes, and thereon alleges, that Defendants have actively induced infringement of at least claim 2 of the '628 patent in violation of 35 U.S.C. § 271(b) by knowingly and intentionally encouraging or aiding third parties to use infringing software and hardware products in this judicial district and elsewhere throughout the United States, without license or authority from SoftView, including at least the Accused Products. SoftView is informed and believes, and thereon alleges, that these third parties have infringed at least claim 2 of the '628 patent in violation of 35 U.S.C. § 271(a) by using infringing software and hardware products, including the Accused Products. The Defendants through at least their user manuals,

Web site support and training materials actively induced their customers and users of the Accused Products to infringe the ’628 patent. Furthermore, the natural and intended use of the Accused Products (*e.g.*, browsing the internet with Samsung Internet and Chrome browsers and viewing HTML email with the Gmail and Android Mail apps) infringes at least claim 2 of the ’628 patent. Defendants know and intend that the Accused Products are used in this infringing manner. In addition, SoftView reasserts evidence presented in Count I *supra* supporting that Defendants knowingly induce infringement of at least claim 2 of the ’628 patent.

343. Defendants have induced infringement of at least claim 2 of the ’628 patent.

344. On information and belief, Defendants have committed acts of infringement notwithstanding actual knowledge of the ’628 patent and without a good faith basis to believe that their activities do not infringe at least claim 2 of the ’628 patent. All infringement of the ’628 patent following Defendants’ knowledge of the ’628 patent is willful, and SoftView is entitled to treble damages and attorneys’ fees and costs incurred in this action under 35 U.S.C. §§ 284 and 285.

345. In the event Defendants assert they were not aware of the ’628 patent, or lack the requisite knowledge and intent to induce infringement, Defendants would have been willfully blind of SoftView’s patent rights.

346. In view of the Delaware litigation and multiple patent applications pending and that issued (including the ’628 patent) during the litigation from the same family as the ’353 and ’926 patents, along with three separate infringement notice letters sent to Defendants, Defendants were aware of a high probability of the existence of such applications and patents and infringement, and/or would had to have taken deliberate actions to avoid learning of the existence of such applications and patents and infringement, including the ’628 patent.

347. Defendants' actions are at least objectively reckless as to the risk of infringement and this objective risk was either known or should have been known by Defendants.

348. Defendants' direct and indirect infringement of the Asserted Patents is and has been willful, intentional, deliberate, and/or in conscious disregard of SoftView's rights under the Asserted Patents.

349. SoftView has incurred substantial damages, including monetary damages.

350. SoftView has been irreparably harmed by Defendants' infringement of the '628 patent.

351. Therefore, SoftView is entitled to actual and/or compensatory damages, reasonable royalties, pre-judgment and post-judgment interest, enhanced damages, and costs.

COUNT V: INFRINGEMENT OF U.S. PATENT NO. 7,461,353

352. SoftView incorporates by reference and re-alleges the foregoing paragraphs of this Complaint as if fully set forth herein.

353. Defendants have directly infringed at least claim 209 of the '353 patent.

354. As a matter of law, the notice provisions of 35 U.S.C. § 287 do not apply in this case for the '353 patent, and therefore SoftView has satisfied its obligations under 35 U.S.C. § 287 with respect to the '353 patent. In addition, SoftView satisfied its obligations under 35 U.S.C. § 287 as to the '353 patent by suing Defendants for infringement in a prior lawsuit in Delaware. Defendants were served with the Second Amended Complaint in that action on October 5, 2011.

355. At the time of the claimed invention, reproducing web pages on a variety of devices presented a significant technical problem. Most internet content displayed as a flat single resolution with no browser support for zoom. '353 Patent, 2:1-3. Much of the content on the internet was designed for display on desktop computers with a single target resolution (e.g., 640x480, 800x600). '353 Patent, 2:4-5. Major internet content providers chose to create their

Web pages using fixed resolution structures, such as tables, which gave them the ability to control the look and feel of their Web sites. '353 Patent, 2:5-10. The fixed resolution approach was so ubiquitous that it became the most common method to brand or uniquely identify websites. '353 Patent, 2:10-13.

356. While the fixed resolution approach was useful for site branding and product differentiation, it presented a daunting technical problem: how to display Internet content designed for desktop computers on devices with smaller screens, lower resolution, and/or different aspect ratios, such as cell phones and handheld computers. '353 Patent, 2:13-18. For example, a website designed for the fixed resolution of a larger desktop display would not fit correctly on small screens, low resolution screens, or devices with different aspect ratios.

357. The '353 patent claimed invention provides a solution to this daunting technical problem. Claim 209 recites as follows:

> 209. A method, comprising:
> rendering a browser interface on a display via which a user of a device is enabled to request access to a Web page, the Web page comprising HTML-based Web content having an original format including HTML code defining an original page layout, functionality, and design of corresponding content on the Web page;
> retrieving the Web page, via the device, and translating at least a portion of the HTML-based Web content into scalable content that supports a scalable resolution-independent representation of the Web page that preserves the original page layout, functionality and design of the content defined by its original format when scaled and rendered; and
> employing the scalable content to render the Web page on the display using a first scale factor; and
> enabling the Web page to be displayed at a different resolution by scaling the scalable content using a second scale factor to re-render the display,
> wherein the original page layout, functionality, and design of the Web page content are preserved under both the first and second scale factors.

358. As claimed in claim 209 of the '353 patent, HTML-based Web content is translated into scalable content, which allows a scalable resolution-independent representation of the Web

page that preserves the original page layout, functionality, and design of the content defined by its original format. The Web page can then be displayed at multiple resolutions by applying scale factors while preserving the original page layout, functionality, and design of the Web page content.

359. Claim 209 of the '353 patent is directed to a technological solution to a technological problem, and was not well-known, routine, or conventional.

360. The advantages of the '353 patent's claimed technical solution are substantial. Web pages could now "be rapidly rendered, zoomed, and panned. Moreover, the rendered displays provide substantially the same or identical layout as the original Web page, enabling users to easily navigate to selected content and features on familiar Web pages." '353 Patent, Abstract. This meant that users could finally be presented with web pages that fit and/or were effectively usable on devices with smaller screens, different resolutions, or different aspect ratios.

361. With the claimed invention, the web page could be presented as it would be on a desktop computer (as most web pages were designed for), and then rapidly panned and zoomed to facilitate use on devices with smaller screens, different resolutions, or different aspect ratios. A Web page that was unusable when presented with the fixed resolution approach could now be zoomed to suit the device display. Today, this functionality is pervasive on smartphone and tablet devices and taken for granted. But prior to the claimed invention, it did not exist.

362. In addition, the claimed invention allowed the look and feel of the Web page to be maintained, even when designed for fixed resolutions. This allowed content providers to continue branding their Web pages with fixed resolution designs. It also allowed users to access Web pages as they were used to seeing them on desktop computers suited to their fixed resolution design,

which meant users knew where to locate items on the page based on its familiar look and feel. Thus, the claimed invention made web pages more convenient for users.

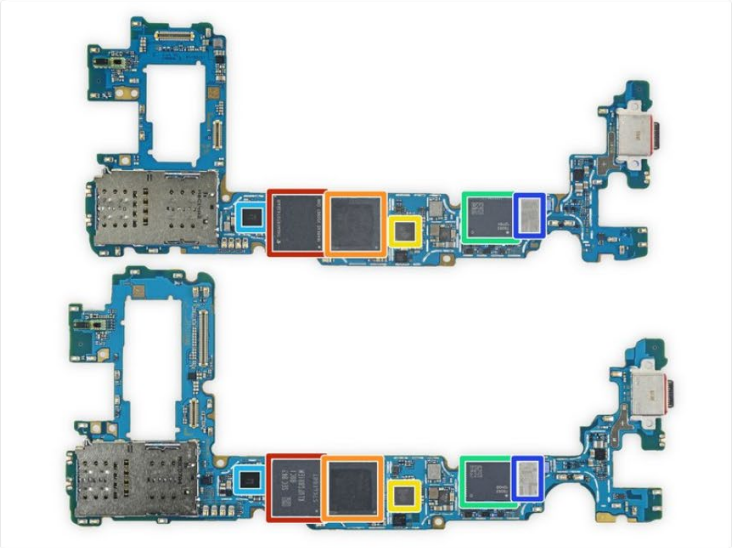
363. The claimed invention has an additional advantage: internet content providers can rely on the fixed resolution approach instead of designing numerous different versions of the same web page to suit the screen size, resolution, and aspect ratio of different devices. For example, a web page designed for a 16-inch desktop display could be effectively viewed and used on a small hand-held mobile phone. This is superior to the alternative approach of designing multiple versions of the same web page to suit different devices, which was time consuming, costly, and impractical for many Web content developers.

364. Defendants have directly infringed (literally and/or under the doctrine of equivalents) at least claim 209 of the '353 patent in violation of 35 U.S.C. § 271 *et seq*., by making, using, offering for sale, selling in the United States, and/or importing into the United States without authority or license the Accused Products.

365. Claim 209 of the '353 Patent recites: "A method, comprising." To the extent that this preamble is limiting, the Accused Products perform a method as further described below.

366. The Accused Products include hardware and software for performing the claimed methods. This includes, for example, a processor, a wireless communications device, a touch-sensitive display (*e.g.*, touchscreen display), and memory/storage used to store instructions including the Android operating system and applications that run on the device, which include the Samsung Internet browser and the Android Chrome browser. For example, evidence of such components is publicly available via teardowns of some of the Accused products that have been published on the Internet, such as presented below for the Samsung Galaxy S10 smartphone

(available at https://www.ifixit.com/Teardown/Samsung+Galaxy+S10+and+S10e+Teardown/120331):



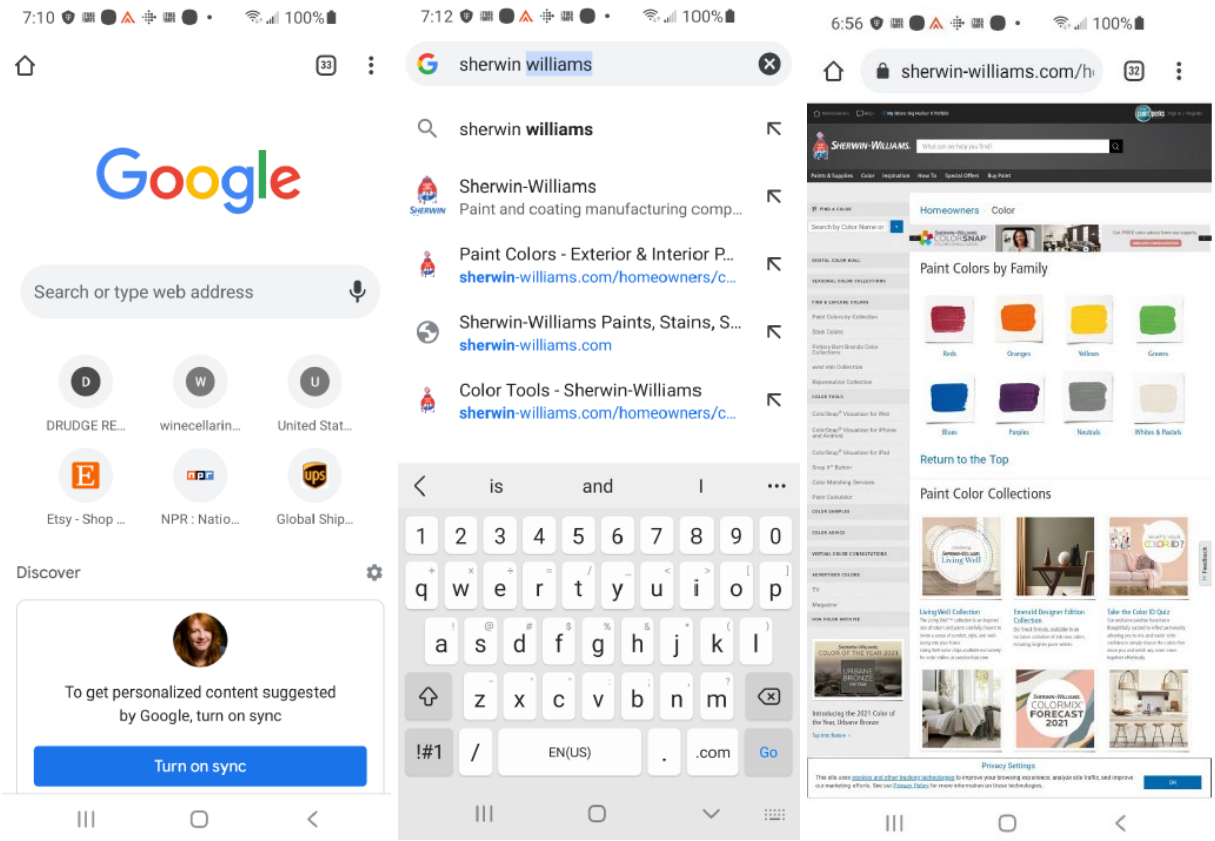
- Thermal pads and cameras aside, let's get to those chips! On the front side of these motherboards (top: S10e, bottom: S10), we spot:
  - S10e: THGAF8T0T43BAIR 128 GB Toshiba UFS NAND flash storage
  - S10: KLUFG8R1EM-B0C1 512 GB Samsung eUFS NAND flash storage
  - Samsung K3UH7H70AM-AGCL LPDDR4X layered over Qualcomm Snapdragon 855 SoC
  - Qualcomm WCD9341 audio codec
  - Qorvo QM78062, likely a RF Fusion front-end module
  - Maxim MAX77705C PMIC
  - Skyworks SKY78160-51 Low Noise Amplifier

Evidence of use of other components are provided via the Defendant's Website (*e.g.*, www.samsung.com), as well as third-party Websites including www.gsmarena.com and www.phonedb.net.

367. Claim 209 of the '353 Patent recites: "rendering a browser interface on a display via which a user of a device is enabled to request access to a Web page, the Web page comprising HTML-based Web content having an original format including HTML code defining an original page layout, functionality, and design of corresponding content on the Web page." The Accused Products perform this operation.

368. Each of the Samsung Internet browser and the Android Chrome browser provide a browser interface on a display (*e.g.*, touchscreen) with multiple means for enabling a user to request access to a Web page comprising HTML-based Web content having an original format including HTML code defining an original page layout, functionality, and design of corresponding content on the Web page. For example, a user may select to open a new browser tab in which the

user may enter a URL for a Web page (*e.g.*, www.nytimes.com) or enter a search query. In response to entry of the search query, the browsers will present results, or the user may select a full search (such as a Google search). The hyperlinks in the search results may be used to launch new Web pages. The screenshots below illustrate an example of this process.



369. HTML (Hypertext Markup Language) and Cascading Style Sheets (CSS) employ standardized specifications for how a Web page is to appear and function. HTML is the standard markup language for documents designed to be displayed in a web browser. HTML describes the structure of a web page semantically and originally included cues for the appearance of the document. Cascading Style Sheets (CSS) is a style sheet language used for describing the presentation of a document written in a markup language such as HTML. CSS is designed to enable the separation of presentation and content, including layout, colors, and fonts. The original

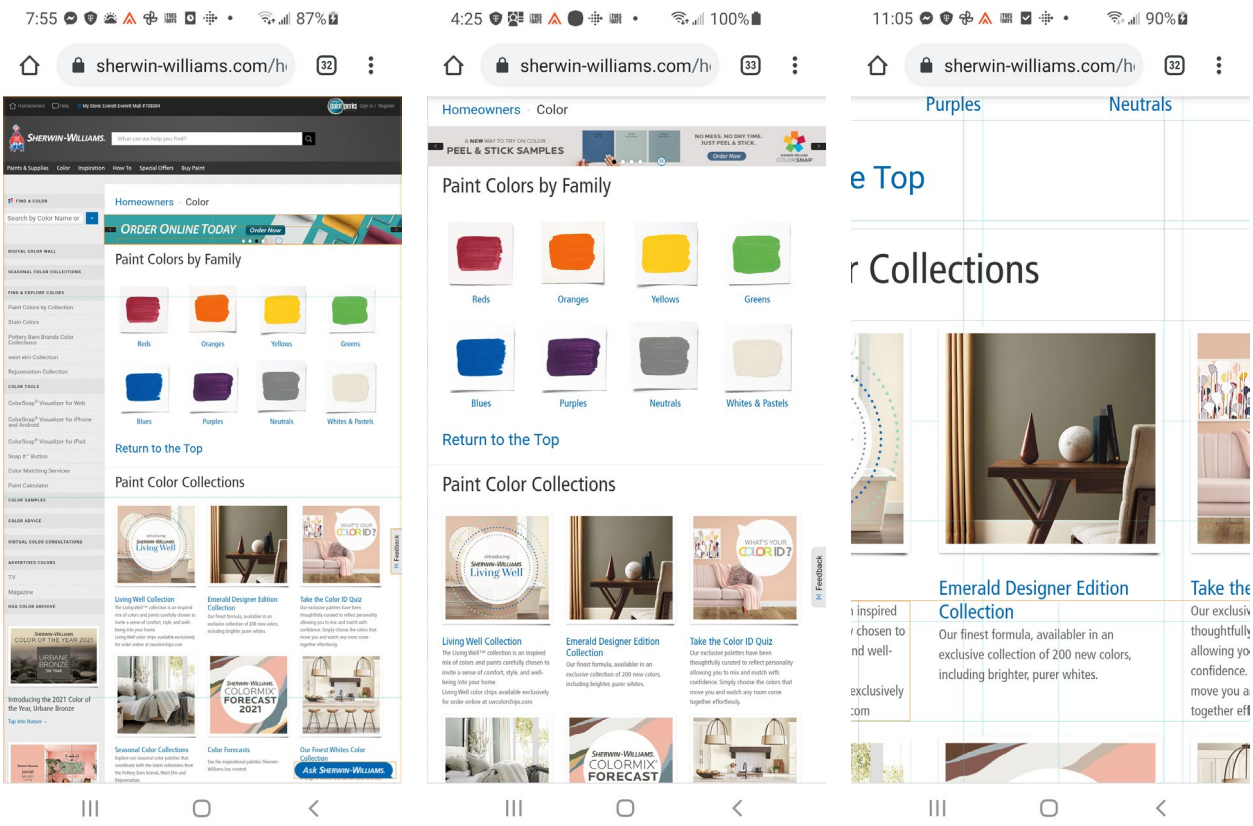
page layout, functionality, and design of content of the Web page is defined by the HTML-based Web content.

370. The layout, functionality, and design of Web pages are interpreted by a browser (*e.g.*, Samsung Internet, Chrome) "rendering engine," also referred to as a "layout engine" or "browser engine." The rendering engine handles layout and rendering of page content. In addition, the rendering engine handles navigation through hyperlinks and data submitted through forms, and implements the Document Object Model (DOM) data structure exposed to page scripts.

371. Claim 209 of the '353 patent recites:

> retrieving the Web page, via the device, and translating at least a portion of the HTML-based Web content into scalable content that supports a scalable resolution-independent representation of the Web page that preserves the original page layout, functionality and design of the content defined by its original format when scaled and rendered; and
>
> employing the scalable content to render the Web page on the display using a first scale factor;

372. The Accused Products perform these operations. The screenshots below demonstrate the Galaxy S10 Accused Product in the Chrome browser employing the scalable content to render the Web page on the display using a first scale factor (first image). Two additional scale factors are also shown (second and third images).



373. The Android operating system includes libraries and frameworks associated with the "WebKit" rendering engine. In 2013, Google began working on a fork of WebKit called "Blink," which is the version of WebKit used in Android and by both the Android Internet browser and the Android Chrome browser. Both WebKit and Blink are HTML- and CSS-compliant rendering engines (meaning they comply with applicable HTML and CSS standards), and thus render Web pages having an original page layout, functionality, and design as defined by the page's HTML-based Web content. Although Blink is a separate fork, much of the core code (which is part of WebCore) is the same for both WebKit and Blink.

374. WebKit/Blink in the Samsung Internet and Chrome browsers process the HTML-based content to generate a DOM and lay out page content (defined by DOM nodes) using bounding boxes as defined by the CSS box model.

375. Each HTML element in the DOM, such as an image, paragraph, header, *etc.*, has an associated box in which the content associated with the HTML element is rendered. There are also HTML elements used for page layout such as divisions (, a container for a *block* of content) and tables (and associated elements). Elements within a are children of the .

376. Illustrations of WebKit's main flow and the CSS box model are shown below. Many of the core rendering processes employed by WebKit and Blink are similar as Blink and WebKit share the same core rendering code (WebCore). Blink uses a separate CSS layout engine called LayoutNG and uses a separate JavaScript engine. Aspects of the main flow for Blink are similar to the main flow for WebKit below, including use of HTML and CSS parsers and generation of a Document Object Model (DOM) and DOM Tree.

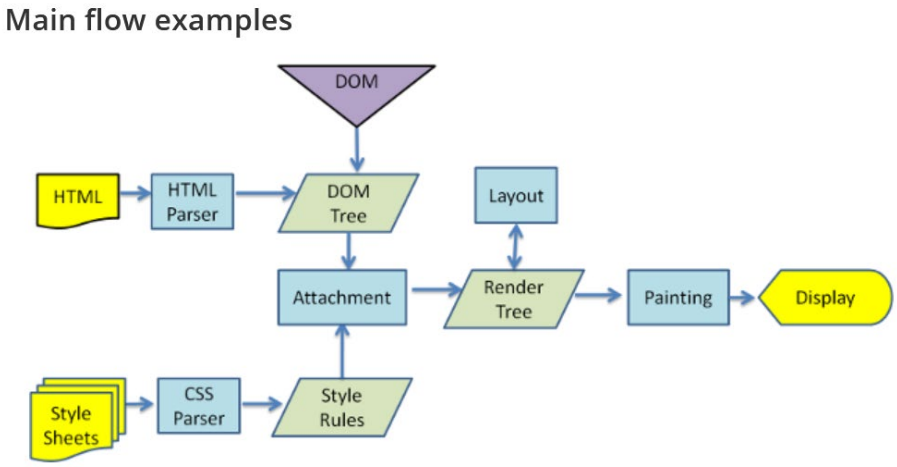


Figure : WebKit main flow

CSS Box model

The CSS box model describes the rectangular boxes that are generated for elements in the document tree and laid out according to the visual formatting model.
Each box has a content area (e.g. text, an image, etc.) and optional surrounding padding, border, and margin areas.

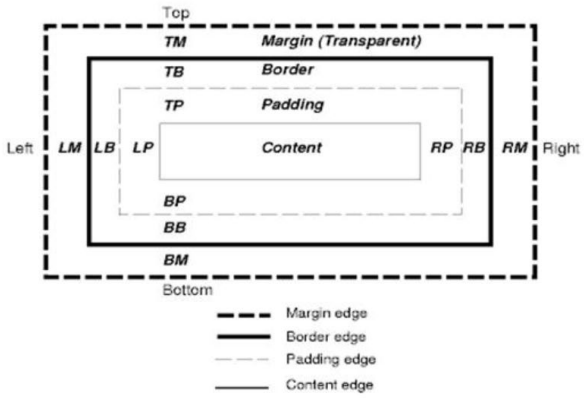


Figure : CSS2 box model

https://www.html5rocks.com/en/tutorials/internals/howbrowserswork/

377. LayoutNG uses Logical & Physical Units.

Logical & Physical Units

To aid in readability and to ensure proper conversion between coordinate systems a set of dedicated units will be used to represent offsets, locations, sizes, and rects for each coordinate system where they are applicable. Conversion between coordinate systems must be done explicitly and no implicit conversion will be allowed.

Logical Units

NGLogicalOffset and NGLogicalSize represents an offset or a size. Each has a pair of LayoutUnits [11] that

represents the offset or size in the inline and block directions.

NGLogicalRect represents a rectangle and contains an offset and a size, represented by the two units described above.

These are all in a logical coordinate space in the sense that they do not take writing mode or directionality into account.

Physical Units

NGPhysicalOffset, NGPhysicalLocation, and NGPhysicalSize represents an offset, location or a size. The location unit is different from the offset one in that it represents the resolved location from the top of the tree while the offset unit represents the delta from the parent. Each has a pair of LayoutUnits that represents the offset or size in the horizontal and vertical directions.

NGPhysicalRect represents a rectangle and contains an offset and a size, represented by the two units described above.

These are all in a physical coordinate space in the sense that they represent coordinates in a top-to-bottom, left-to-right coordinate system. Converting from a logical unit to a physical one thus requires the coordinates to be resolved. The *Writing Modes and Coordinate Systems* section goes into more details.

https://docs.google.com/document/d/1uxbDh4uONFQOiGuiumlJBLGgO4KDWB8ZEkp7Rd47fw4/edit?tab=t.0#heading=h.guvbepjyp0oj.

378. LayoutNG generates a Layout Tree comprising NGFragment and NGText Nodes.

Layout Tree

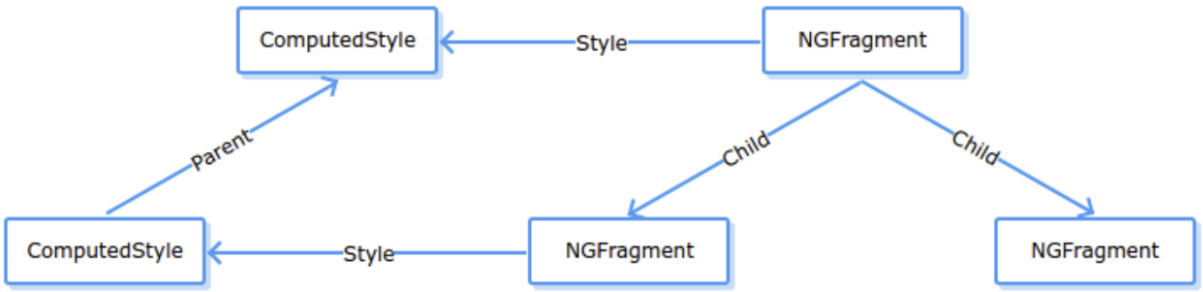


Figure shows the style tree (left) and layout tree (right) and their relationship.

The layout tree is made up entirely of NGFragment and NGText nodes and is immutable. Each node includes a back pointer to the computed style, a list of children and the size and location of the node itself. The location is relative to the immediate parent.

Given the immutability of the tree, use of relative coordinates, and the lack of a parent pointer entire subtrees may be reused across layouts. This concept is discussed in detail in the subsequent layouts section.

https://developer.chrome.com/docs/chromium/blinkng.

379. The Layout Tree also represents the bounding box hierarchy of the HTML document.

Pipelining style, layout, and pre-paint

Collectively, the rendering phases before *paint* are responsible for the following:

- Running the *style cascade* algorithm to calculate final style properties for DOM nodes.
- Generating the layout tree representing the box hierarchy of the document.
- Determining size and position information for all boxes.
- Rounding or snapping sub-pixel geometry to whole pixel boundaries for painting.
- Determining the properties of composited layers (affine transformation, filters, opacity, or anything else that can be GPU accelerated).
- Determining what content has changed since the previous paint phase, and needs to be painted or repainted (paint invalidation).

https://developer.chrome.com/docs/chromium/blinkng.

380. LayoutNG uses ClientRects to Size and position HTML elements (DOM objects) relative to the viewport.

ClientRect	Size and position of an Element relative to the viewport.

Walking up the Tree

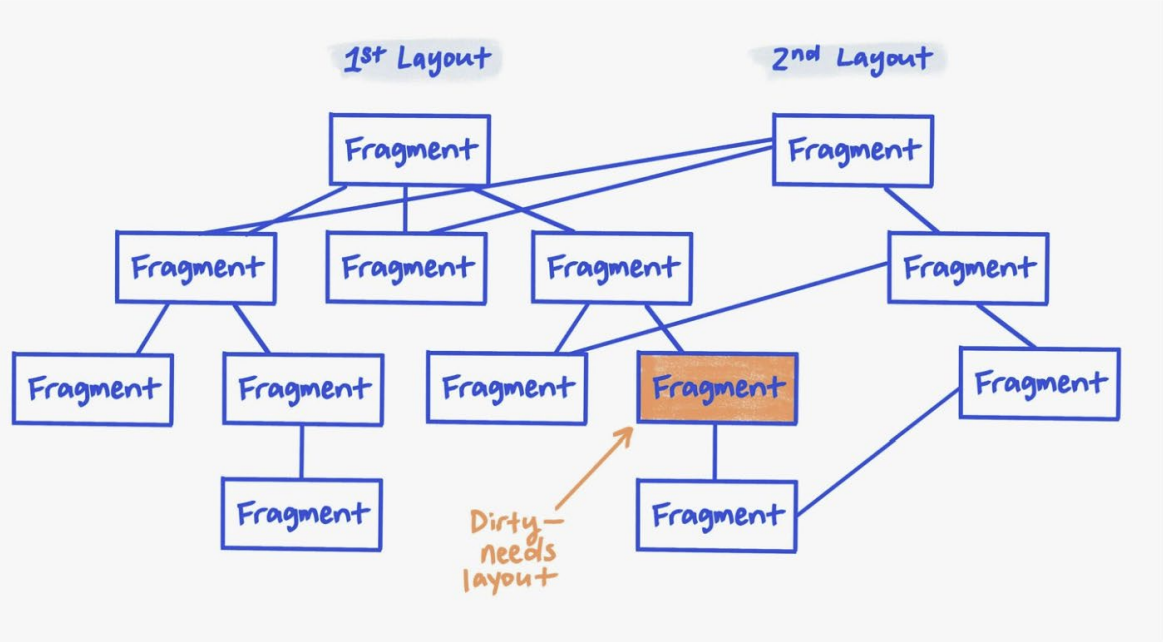
ClientRect calculations and IntersectionObserver need to perform a tree-walk up to the root in order to apply transform/layout information. Transform information is not on the layout tree. Adding layout-parent pointers in the Layout Tree would mean that we lose the immutability property; so instead the parent-layout pointers will be stored on the DOM tree. These operations will begin at the DOM node, ask its NGUnplacedFragment for layout information, then walk up the DOM tree to its parent-layout and so forth.

https://docs.google.com/document/d/1uxbDh4uONFQOiGuiumlJBLGgO4KDWB8ZEkp7Rd47fw4/edit?tab=t.0#heading=h.guvbepjyp0oj.

381. Blink generates an immutable fragment tree, which is used to represent the position and size of all elements on the page (without transforms applied).

The immutable fragment tree

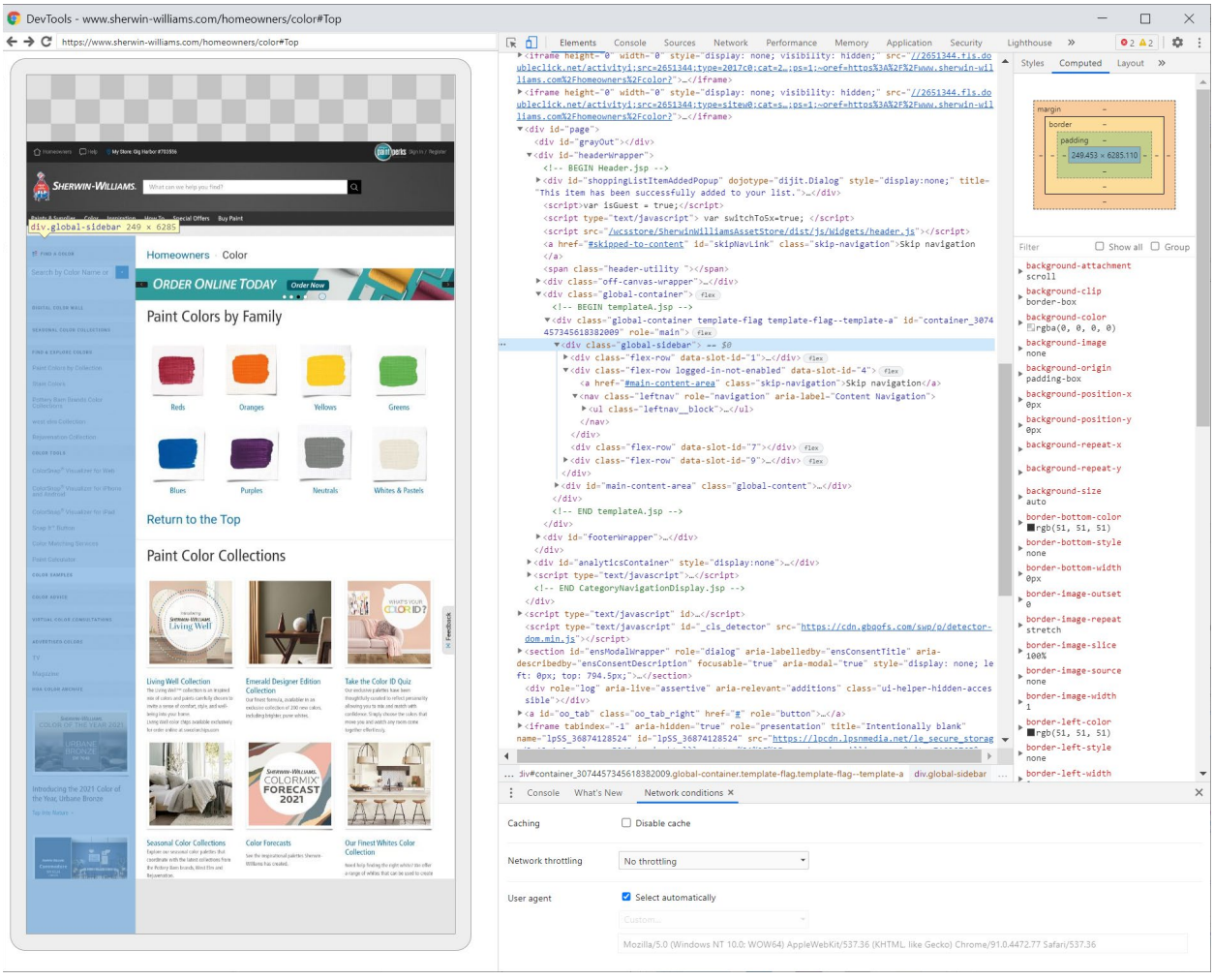
The immutable fragment tree is the output of the layout stage of the rendering pipeline. It represents the position and size of all elements on the page (without transforms applied).



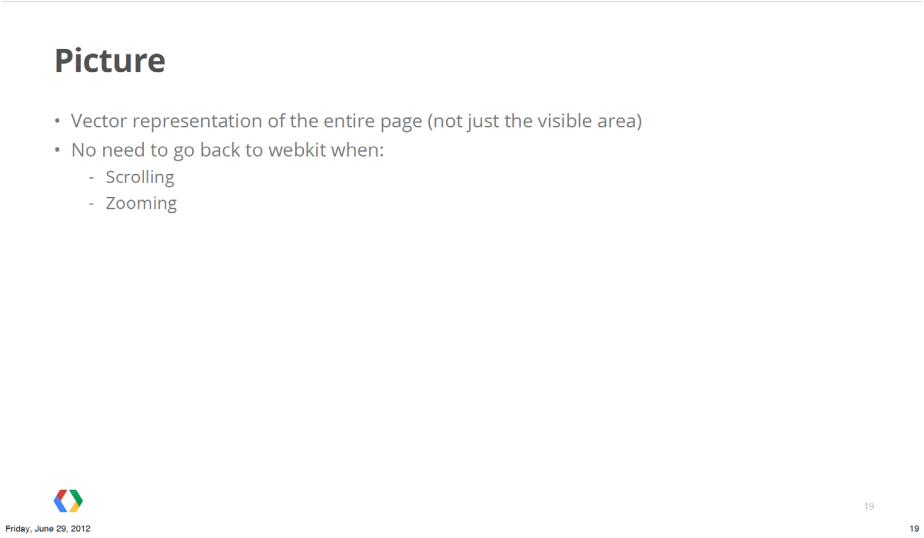
Each fragment represents a part of a DOM element. Typically there is only one fragment per element, but there can be more if it is split across different pages when printing, or columns when in a multi-column context.

https://developer.chrome.com/docs/chromium/renderingng-data-structures.

382. The following shows example layout information for a Web page being browsed on a Samsung Galaxy S10 using the Chrome Web Inspector, which is running on a laptop to which the S10 is connected via a USB-C cable. The Web page was retrieved via the Samsung Galaxy S10 device using the Chrome browser. This example shows the layout box for the left-hand column called "global_sidebar" is 249 x 6285 LayoutUnits, in accordance with the CSS box model. There is a similar layout box for each HTML element on the page.

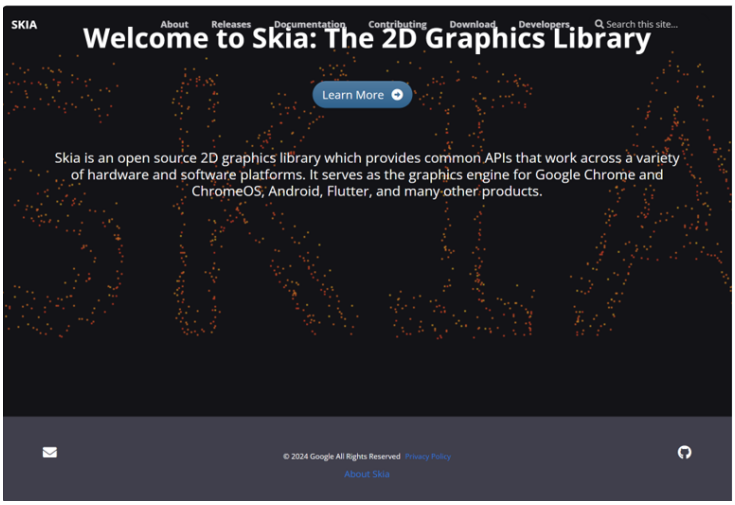


383. WebView converts the page layout information into a scalable "vector representation" of the entire page that supports a scalable resolution-independent representation of the Web page. This supports scrolling and zooming.



Android WebView — Google IO 2012.

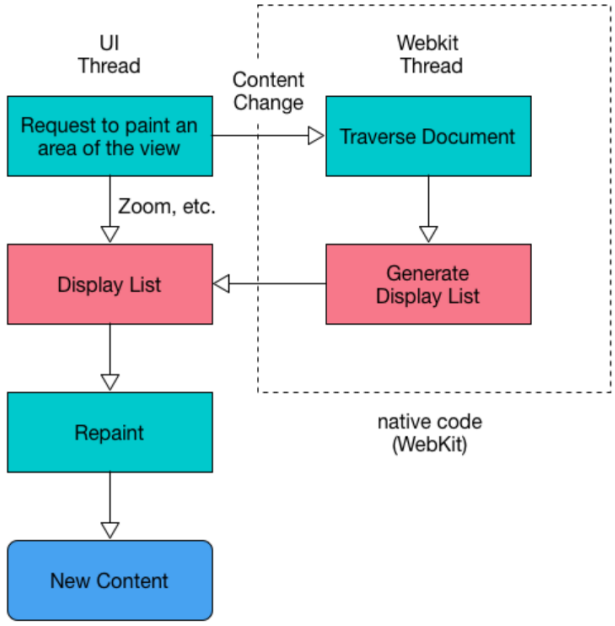
384. The Samsung Internet and Chrome browsers use the Skia 2D vector graphics rendering software/library, which is also referred to as the Skia Graphics Engine.



385. WebView (and the Samsung Internet and Chrome browsers) records lists of painting instructions for the full page and builds display lists. Zooming is accomplished by changing scale and translation parameters in a transformation matrix (SkMatrix) to scale and/or

translate position vector coordinates and calling the painting instructions in the display list. While the following describes the approach used by the Chrome browser with WebKit, the current painting process using Blink is similar.

> So, instead of directly asking webkit to *paint* the screen, the team had a better idea. They decided to modify Webkit to *record* the list of painting instructions for the full page, but to not paint directly — basically using a "display list" model. The list of painting commands were for example "draw this text here", or "draw this image there".

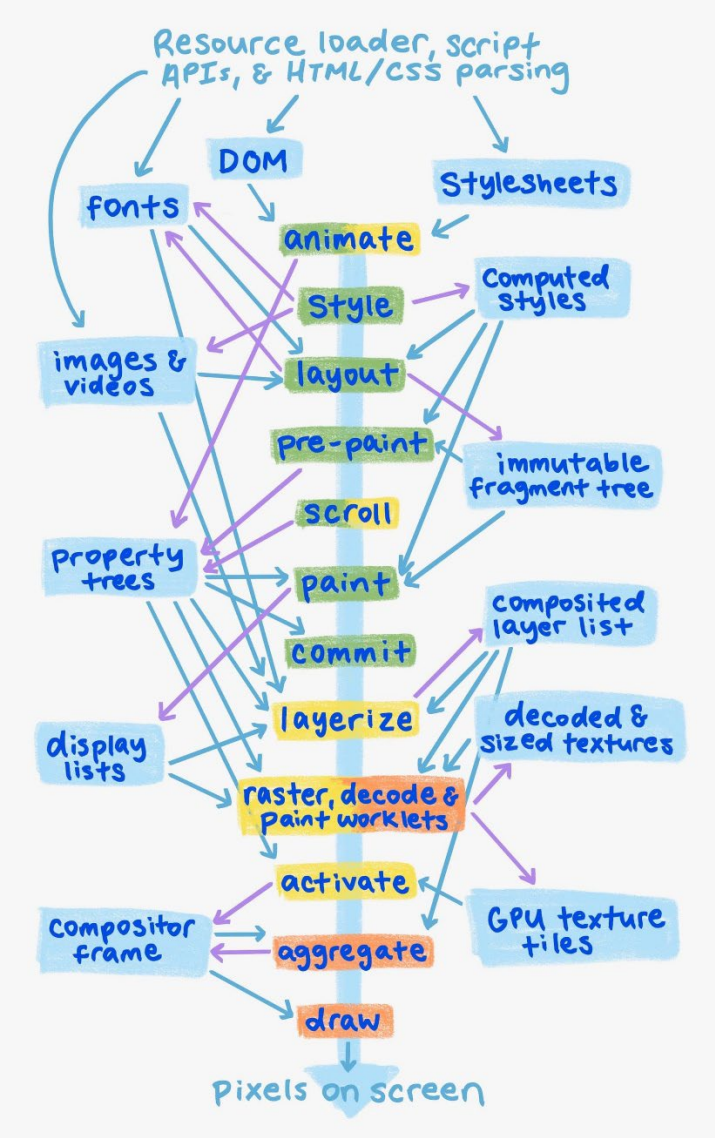


Rendering loop (Android)

> When the user would need to zoom in for example, it was not necessary to ask Webkit to repaint the screen — the webview instead could just replay the already present display list, only changing the zoom factor. This also gave a rather nice side-effect that the screen was always sharp — no text blurriness when zooming in, no missing content, as the content was always being redrawn "the right way".
>
> You could think of this display list as a way to have a vector representation of the entire page, that you could scale on demand and redraw — the difference between illustrator and photoshop.

https://medium.com/@camaelon/what-s-in-a-web-browser-83793b51df6c.

386. Display lists are used in the current Rendering Pipeline.



The stages are:

1. *Animate:* change computed styles and mutate *property trees* over time based on declarative timelines.
2. *Style:* apply CSS to the DOM, and create *computed styles*.
3. *Layout:* determine the size and position of DOM elements on the screen, and create the *immutable fragment tree*.
4. *Pre-paint:* compute property trees and invalidate any existing *display lists* and GPU *texture tiles* as appropriate.
5. *Scroll:* update the scroll offset of documents and scrollable DOM elements, by mutating property trees.
6. *Paint:* compute a display list that describes how to raster GPU texture tiles from the DOM.
7. *Commit:* copy property trees and the display list to the compositor thread.
8. *Layerize:* break up the display list into a *composited layer list* for independent rasterization and animation.
9. *Raster, decode and paint worklets:* turn display lists, encoded images, and paint worklet code, respectively, into GPU texture tiles.
10. *Activate:* create a *compositor frame* representing how to draw and position GPU tiles to the screen, together with any visual effects.
11. *Aggregate:* combine compositor frames from all the visible compositor frames into a single, global compositor frame.
12. *Draw:* execute the aggregated compositor frame on the GPU to create pixels on-screen.

https://developer.chrome.com/docs/chromium/renderingng-architecture#pipeline.

Display lists and paint chunks

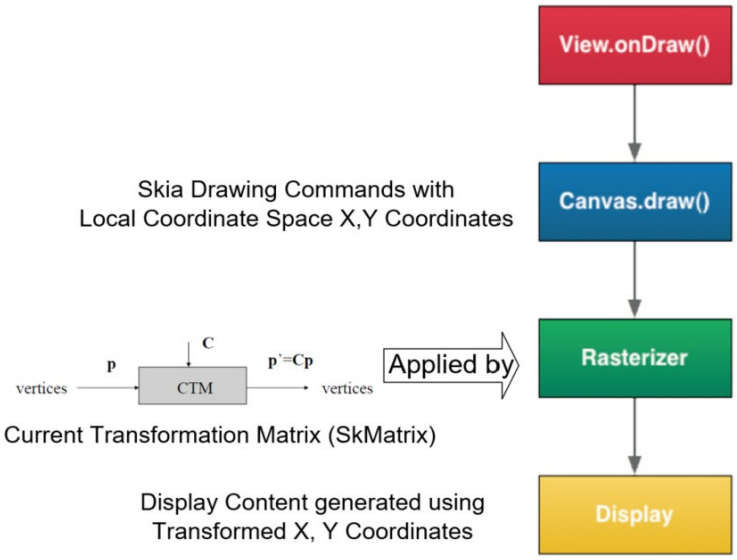
A display item contains low-level drawing commands (see here) that can be rasterized with Skia. Display items are typically simple, with just a few drawing commands, such as drawing a border or background. The paint tree walk iterates over the layout tree and associated fragments following CSS painting order to produce a display item list.

For example:



https://developer.chrome.com/docs/chromium/renderingng-data-structures#display_lists_and_paint_chunks.

387. Skia uses a Current Transformation Matrix (SkMatrix) for scaling and translation of content. An example is shown below for Skia painting.



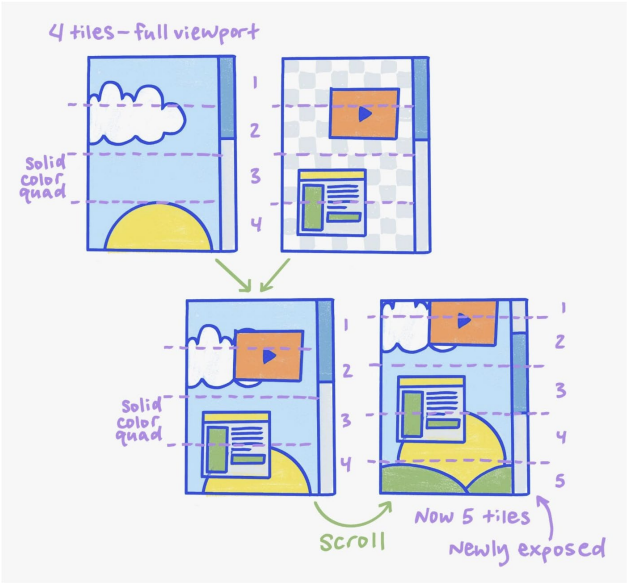
Android Accelerated Rendering Google IO 2011 (modified).

388. The rendering pipeline uses the Skia display lists to rasterize pixels onto GPU texture tiles.

Tiles

In theory, a render process or browser process compositor could rasterize pixels into a single texture the full size of the renderer viewport and submit that texture to Viz. To display it, the display compositor would just have to copy the pixels from that single texture into the appropriate position in the frame buffer (for example, the screen). However, if that compositor wanted to update even a single pixel it would need to re-rasterize the full viewport and submit a new texture to Viz.

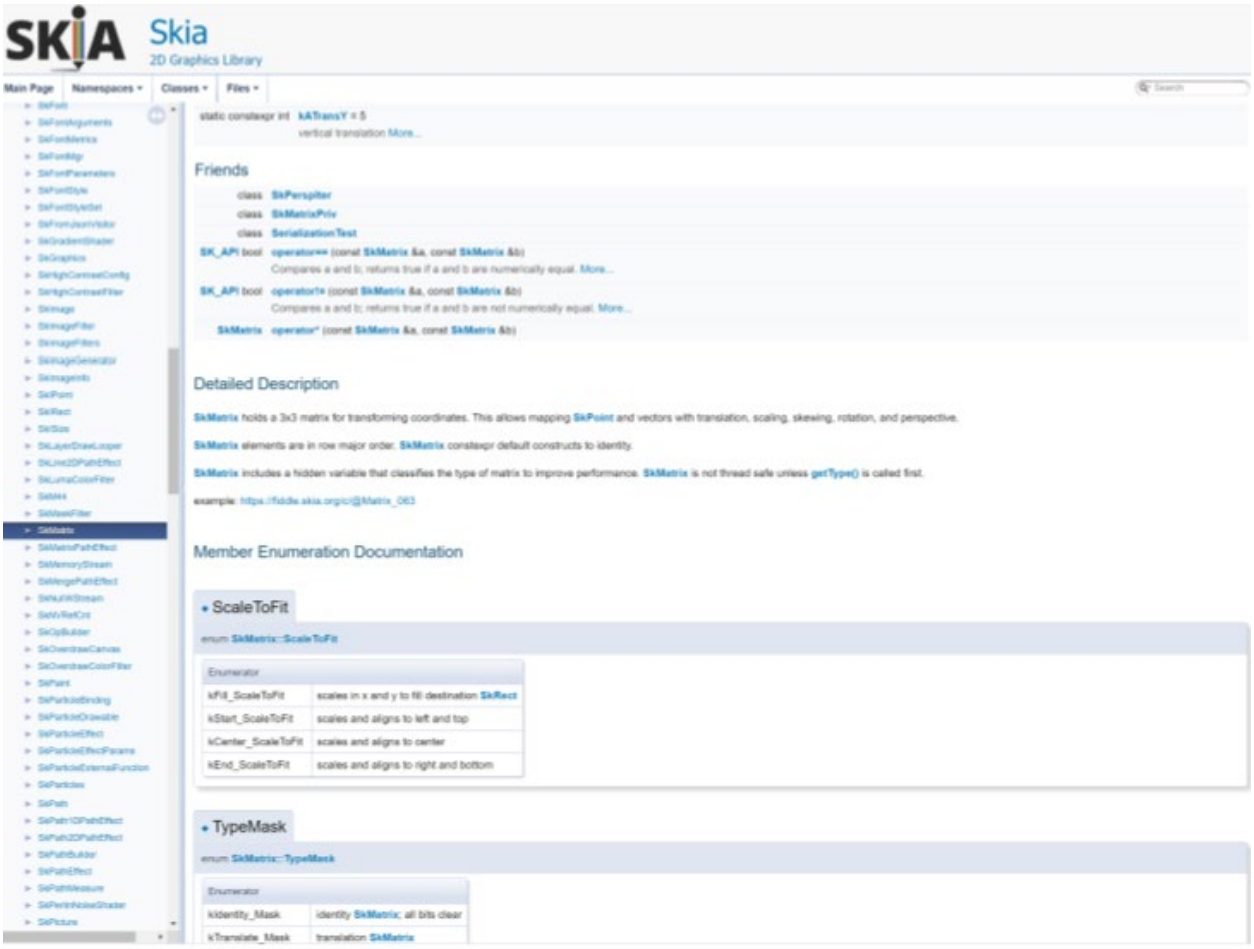
Instead, the viewport is divided into tiles. A separate GPU texture tile backs each tile with the rasterized pixels for part of the viewport. The renderer can then update individual tiles or even just change the position on screen for the existing tiles. For example, when scrolling a website, the position of existing tiles would shift up and only occasionally a new tile would need to be rasterized for content further down the page.



This image depicts an image of a sunny day, with four tiles. When a scroll occurs, a fifth tile begins to appear. One of the tiles happens to only have one color (sky blue), and there is a video and an iframe on top.

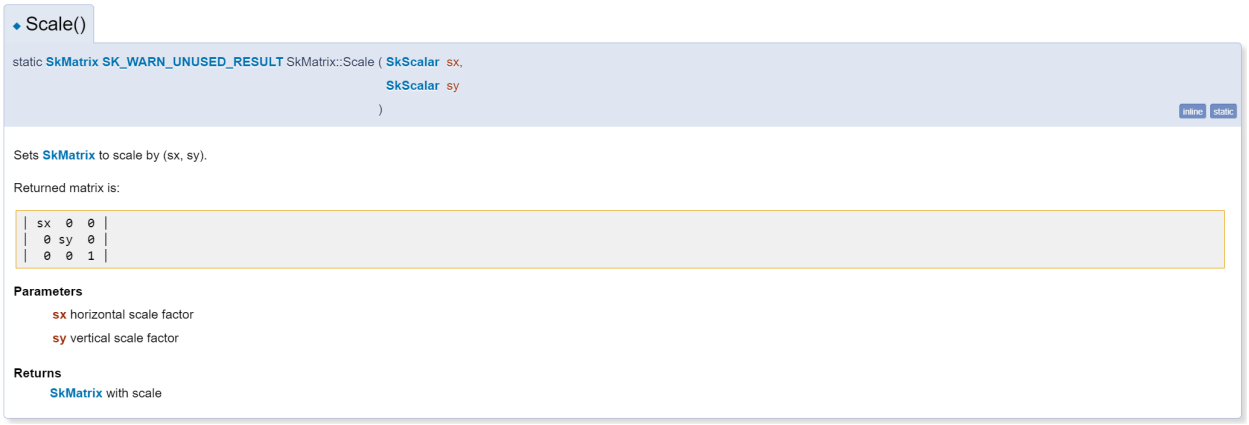
https://developer.chrome.com/docs/chromium/renderingng-data-structures#tiles.

389. The Display List uses the SKMatrix class to effect scaling. SKMatrix holds a 3x3 matrix for transforming coordinates (also known as an affine transformation). SKMatrix is used to effect translation and scaling. Skia renders images by ScaleToFit the destination rectangle (SKRect). SKRect is scaled using the current SKMatrix, and then the Image is scaled to fit the SKRect that is generated.



https://api.skia.org/classSkMatrix.html.

390. The Skia Scale() function can be used to change the sx (horizontal) and sy (vertical) scale factors in the SKMatrix. The setAffine() function can be used to set values for the first two rows of the SKMatrix. This enables scaling and translation to be performed together.



◆ Scale()

static SkMatrix SK_WARN_UNUSED_RESULT SkMatrix::Scale (SkScalar sx,
SkScalar sy
)

inline static

Sets SkMatrix to scale by (sx, sy).

Returned matrix is:

```
| sx  0  0 |
|  0 sy  0 |
|  0  0  1 |
```

Parameters

sx horizontal scale factor

sy vertical scale factor

Returns

SkMatrix with scale



◆ setAffine()

SkMatrix& SkMatrix::setAffine (const SkScalar affine[6])

Sets SkMatrix to affine values, passed in column major order.

Given affine, column, then row, as:

```
| scale-x  skew-x translate-x |
|  skew-y scale-y translate-y |
```

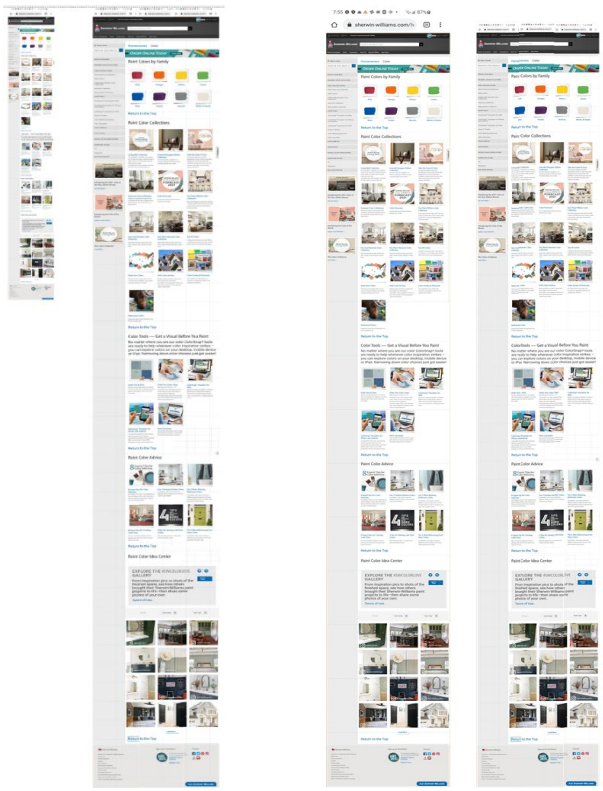
SkMatrix is set, row, then column, to:

```
| scale-x  skew-x translate-x |
|  skew-y scale-y translate-y |
|       0       0           1 |
```

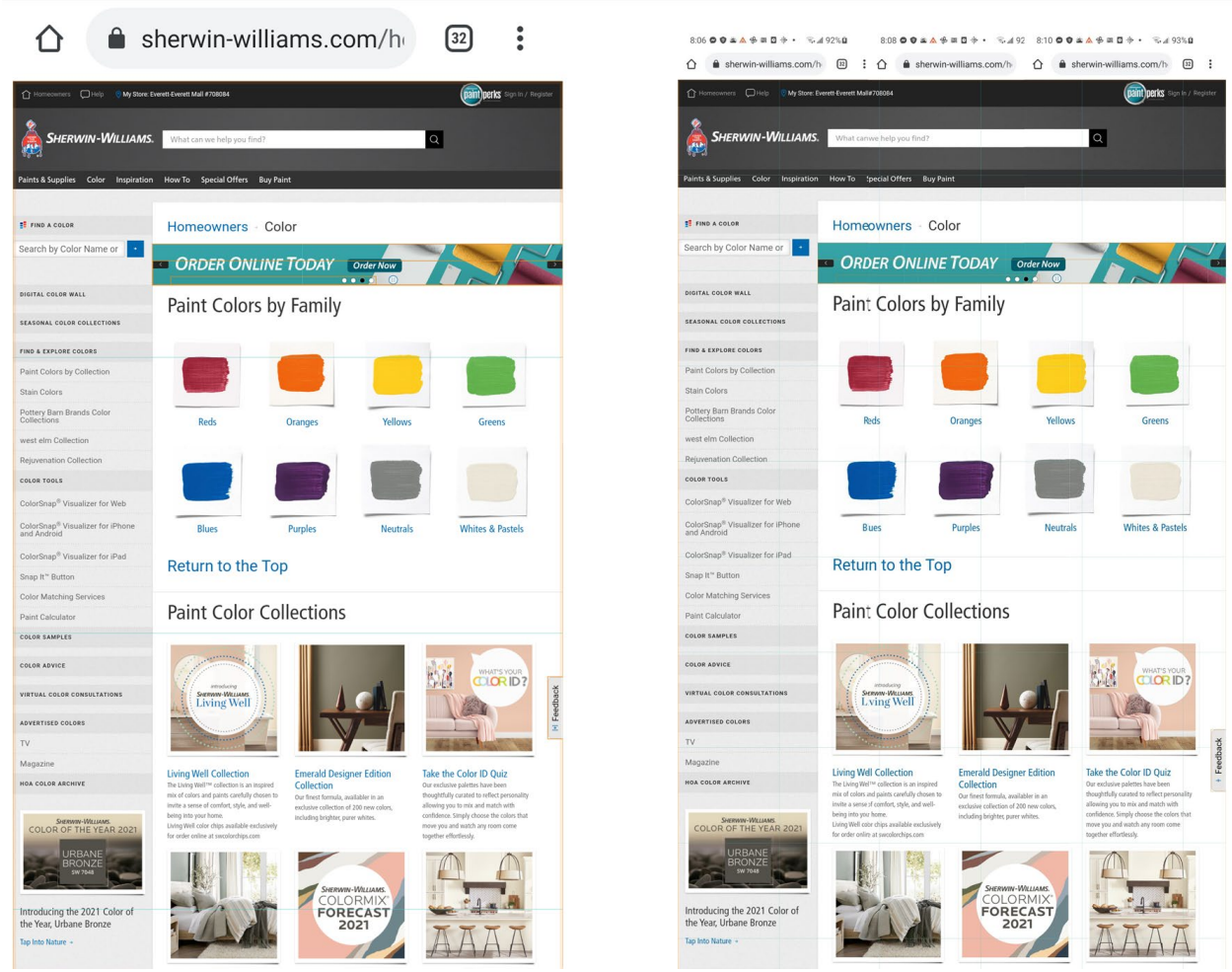
Parameters

affine 3 by 2 affine matrix

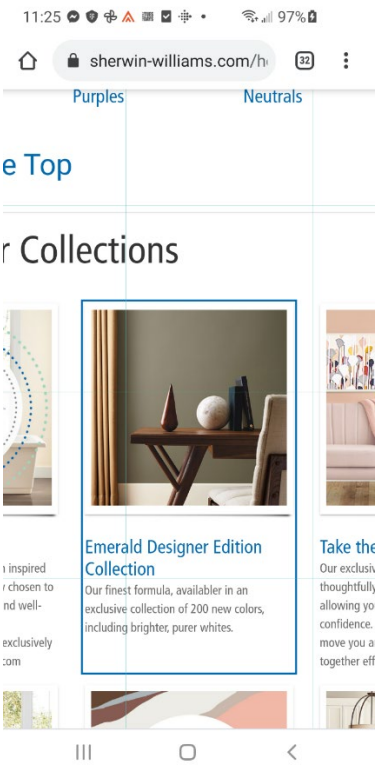
391. The vector representation and use of the scaling (zooming) factors preserves the original page layout, functionality, and design of the Web page content (as defined by the HTML-based Web content). This is demonstrated in the following page. For the first two images, the image on the left is a screenshot mosaic at the original zoom level (corresponding to width of the page when it is loaded), and the image on the right is a screenshot mosaic at a zoom level of approximately 2.4 times the original zoom level. The pair of images on the right show normalized versions of the first two images. The use of vector graphics combined with use of the CSS box model results in perfect scaling under which the original page layout and design is preserved.

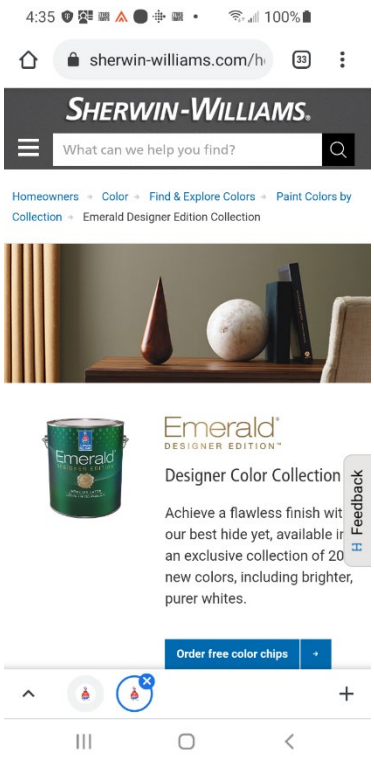


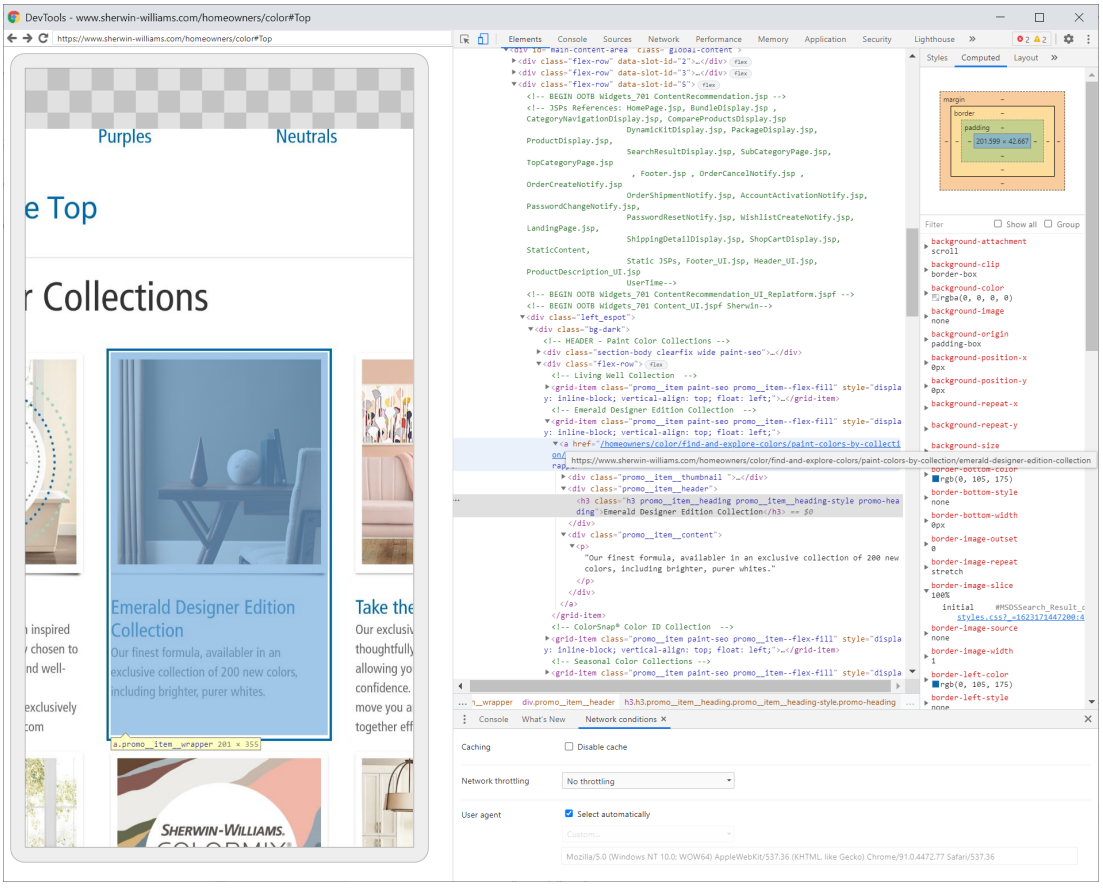
392. The following screenshot (left) and screenshot mosaic (right) show the Web page at its original size (left) and when scaled by approximately 2.4 (right). The cyan lines show the borders for the tiles used to generate the page content.



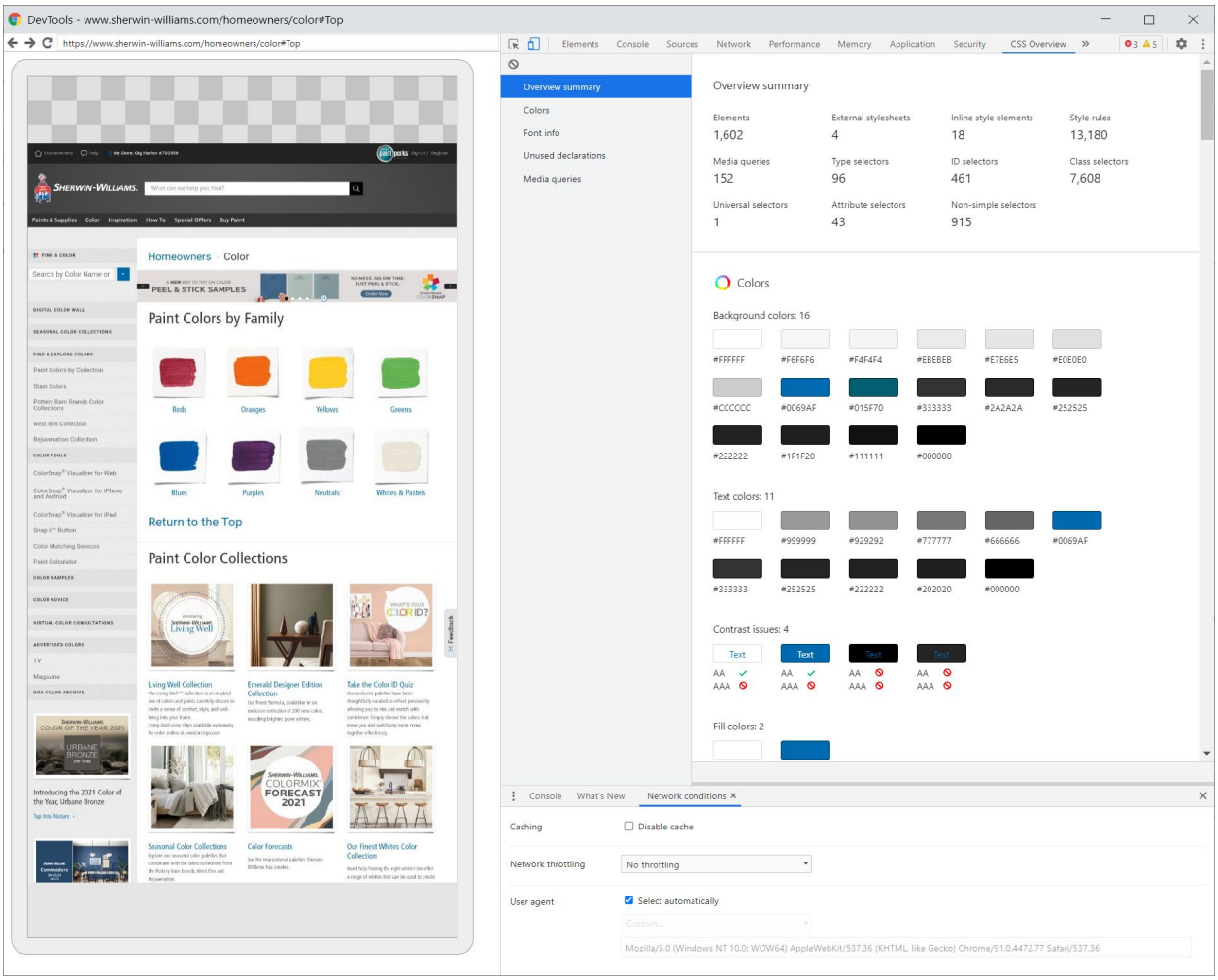
393. The functionality defined by the HTML-based Web content including hyperlink functionality is preserved. This is demonstrated in the following screenshots, where a hyperlink image map in the blue box shown in the page on the left is tapped on, causing the page on the right to load.


11:25
97%
sherwin-williams.com/h
Purples
Neutrals
e Top
r Collections
Emerald Designer Edition Collection
Our finest formula, availabler in an exclusive collection of 200 new colors, including brighter, purer whites.
Take the
Our exclusiv
thoughtfully
allowing yo
confidence.
move you a
together eff


4:35
100%
sherwin-williams.com/h
SHERWIN-WILLIAMS.
What can we help you find?
Homeowners
Color
Find & Explore Colors
Paint Colors by Collection
Emerald Designer Edition Collection
Emerald
DESIGNER EDITION
Designer Color Collection
Achieve a flawless finish wit
our best hide yet, available i
an exclusive collection of 20
new colors, including brighter,
purer whites.
Order free color chips
Feedback


DevTools - www.sherwin-williams.com/homeowners/color#Top
Purples
Neutrals
e Top
r Collections
Emerald Designer Edition Collection
Our finest formula, availabler in an exclusive collection of 200 new colors, including brighter, purer whites.
Take the
Elements
Console
Sources
Network
Performance
Memory
Application
Security
Lighthouse
Styles
Computed
Layout
Network conditions
Caching
Disable cache
Network throttling
No throttling
User agent
Select automatically

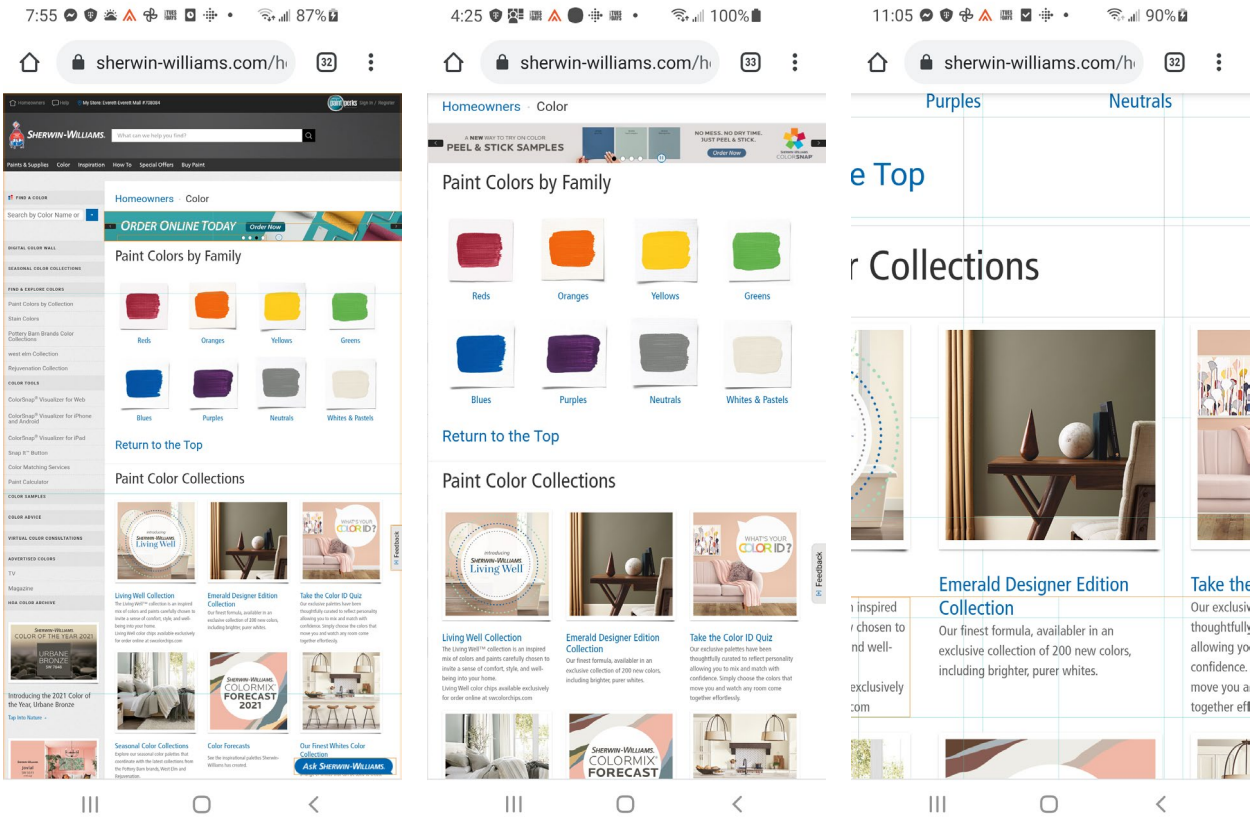
394. This Web page uses CSS. In addition to CSS information for each object, the Chrome Web inspector has a CSS Overview tab that provides some information about CSS usage for the page.



395. Claim 209 recites "enabling the Web page to be displayed at a different resolution by scaling the scalable content using a second scale factor to re-render the display, wherein the original page layout, functionality, and design of the Web page content are preserved under both the first and second scale factors." The Accused Products satisfy these requirements.

396. As shown above, WebView and both the Samsung Internet browser and the Chrome browser enable the Web page to be displayed at a different resolution by scaling the

scalable content using a second scale factor to re-render the display, which preserves the original page layout, functionality, and design of the Web page content (*e.g.*, as shown by screenshots below demonstrating preservation of the original page layout, functionality, and design of the Web page content at different zoom levels).



Samsung's Inducement of Infringement of the '353 Patent

397. Defendants had actual knowledge of the '353 patent based on publicity surrounding the initial filing of the original assertion of the Delaware litigation to which it was added in May 2010. News stories about the litigation and the '353 patent were published on major Web sites including CNET.com, PCMag.com, IP360.com, and TechCrunch.com. These articles identified the '353 patent and provided notice to Defendants that they were infringing the '353 patent.

398. Defendants also had actual knowledge that they were infringing the '353 patent based on being served on October 5, 2011, with the Second Amended Complaint in the Delaware

litigation, which named Defendants as defendants and alleged that Defendants had products that were infringing the ’353 patent.

399. SoftView is informed and believes, and thereon alleges, that Defendants have actively induced infringement of at least claim 209 of the ’353 patent in violation of 35 U.S.C. § 271(b) by knowingly and intentionally encouraging or aiding third parties to use infringing software and hardware products in this judicial district and elsewhere throughout the United States, without license or authority from SoftView, including at least the Accused Products. SoftView is informed and believes, and thereon alleges, that these third parties have infringed at least claim 209 of the ’353 patent in violation of 35 U.S.C. § 271(a) by using infringing software and hardware products, including the Accused Products. The Defendants, through at least their user manuals, Web site support and training materials actively induced their customers and users of the Accused Products to infringe the ’353 patent. Furthermore, the natural and intended use of the Accused Products (*e.g.*, browsing the internet with Samsung Internet and Chrome browsers and viewing HTML email with the Gmail and Android Mail apps) infringes at least claim 209 of the ’353 patent. Defendants know and intend that the Accused Products are used in this infringing manner. In addition, SoftView reasserts evidence presented in Count I *supra* supporting that Defendants knowingly induce infringement of at least claim 209 of the ’353 patent.

400. Defendants have induced infringement of at least claim 209 of the ’353 patent.

401. The infringement of claim 209 of the ’353 patent is willful. Defendants were put on notice of infringement of the ’353 patent during prior litigation (original case, 1:10-cv-00389-LPS filed in District of Delaware to which Defendants were added on September 30, 2011, split out as 1:12-cv-00987-LPS SoftView LLC v. Samsung Electronics Co. Ltd. et al. on July 26, 2012 (see discussion concerning prior litigation *supra*). Prior notice was at least based on being served

on October 5, 2011 with the Second Amended Complaint in Delaware 1:10-cv-00389-LPS. Additional notice was provided through infringement contentions provided to Defendants during the Delaware litigation.

402. By engaging in the conduct described herein, Defendants have injured SoftView and are thus liable for infringement of the ’353 patent, pursuant to 35 U.S.C. § 271.

403. As a result of Defendants’ infringement of the ’353 patent, SoftView has suffered monetary damages and is entitled to a monetary judgment in an amount adequate to compensate for Defendants’ past infringement, together with interests and costs.

404. On information and belief, Defendants have committed acts of infringement notwithstanding actual knowledge of the ’353 patent and without a good faith basis to believe that its activities do not infringe at least claim 209 of the ’353 patent. All infringement of the ’353 patent following Defendants’ knowledge of the ’353 patent is willful, and SoftView is entitled to treble damages and attorneys’ fees and costs incurred in this action under 35 U.S.C. §§ 284 and 285.

405. In the event Defendants assert they were not aware of the ’353 patent, or lack the requisite knowledge and intent to induce infringement, Defendants would have been willfully blind of SoftView’s patent rights.

406. Defendants’ actions are at least objectively reckless as to the risk of infringement and this objective risk was either known or should have been known by Defendants.

407. Defendants’ direct and indirect infringement of the Asserted Patents is and has been willful, intentional, deliberate, and/or in conscious disregard of SoftView’s rights under the Asserted Patents.

408. SoftView has incurred substantial damages, including monetary damages.

409. SoftView has been irreparably harmed by Defendants' infringement of the '353 patent.

410. Therefore, SoftView is entitled to actual and/or compensatory damages, reasonable royalties, pre-judgment and post-judgment interest, enhanced damages, and costs.

PRAYER FOR RELIEF

WHEREFORE, Plaintiff, SoftView, LLC moves this Honorable Court for judgment to be entered in its favor and against Defendant, Samsung Electronics Co., Ltd. and Samsung Electronics America, Inc., and for the following relief:

(i) A declaration that the '729 patent is valid and enforceable;

(ii) A declaration that the '889 patent is valid and enforceable;

(iii) A declaration that the '154 patent is valid and enforceable;

(iv) A declaration that the '353 patent is valid and enforceable;

(v) A declaration that the '628 patent is valid and enforceable;

(vi) A declaration that the Defendants have infringed one or more claims of the '729 patent;

(vii) A declaration that the Defendants have infringed one or more claims of the '889 patent;

(viii) A declaration that the Defendants have infringed one or more claims of the '154 patent;

(ix) A declaration that the Defendants have infringed one or more claims of the '353 patent;

(x) A declaration that the Defendants have infringed one or more claims of the '628 patent;

(xi) An award of damages sufficient to compensate SoftView for Defendants' infringement of the '729 patent, '889 patent, '154 patent, '628 patent, and '353 patent, pursuant to 35 U.S.C. § 284;

(xii) An award of prejudgment and post-judgment interest pursuant to 35 U.S.C. § 284;

(xiii) Treble damages for willful infringement as permitted under 35 U.S.C. § 284;

(xiv) An award of attorneys' fees incurred in prosecuting this action, on the basis that this is an exceptional case provided by 35 U.S.C. § 285; and

(xv) Such other and further relief as this Court shall deem appropriate.

JURY DEMAND

Plaintiff demands a trial by jury of any and all issues triable of right before a jury pursuant to Rule 38 of the Federal Rules of Civil Procedure.

RESERVATION OF RIGHTS

Plaintiff's investigation is ongoing, and certain material information remains in the sole possession of the Defendants or third parties, which will be obtained via discovery herein. Plaintiff expressly reserves the right to amend or supplement the causes of action set forth herein in accordance with Rule 15 of the Federal Rules of Civil Procedure.

DATED: February 28, 2025

Respectfully submitted,

/s/ Bradley W. Caldwell

Bradley W. Caldwell
Texas Bar No. 24040630
Email: bcaldwell@caldwellcc.com
Jason D. Cassady
Texas Bar No. 24045625
Email: jcassady@caldwellcc.com
Justin T. Nemunaitis
Texas Bar No. 24065815
Email: jnemunaitis@caldwellcc.com
Andrew Langford
Texas Bar No. 24087886
Email: alangford@caldwellcc.com
Aisha M. Haley
Texas Bar No. 24139895
Email: ahaley@caldwellcc.com
James F. Smith
Texas Bar No. 24129800
Email: jsmith@caldwellcc.com
CALDWELL CASSADY & CURRY P.C.
2121 N Pearl Street, Suite 1200
Dallas, Texas 75201
Telephone: (214) 888-4848
Facsimile: (214) 888-4849

Andrea L. Fair
Texas Bar No. 24078488
Email: andrea@millerfairhenry.com
MILLER FAIR HENRY, , PLLC
1507 Bill Owens Parkway
Longview, Texas 75604
Telephone: (903) 757-6400
Facsimile: (903) 757-2323

ATTORNEYS FOR PLAINTIFF SOFTVIEW, LLC