XRAY: Inspector Tools For Designers

Leave Authors Anonymous
for Submission
City, Country
e-mail address

Leave Authors Anonymous
for Submission
City, Country
e-mail address

Leave Authors Anonymous
for Submission
City, Country
e-mail address

ABSTRACT

Numerous tools have been created to facilitate developers' borrowing of code from live websites and design examples. However, little work has been done to improve the experience of designers working on partially-developed or live sites. This paper introduces XRAY, inspector tools for designers (Figure 1). Unlike traditional inspector tools, XRAY allows the designer to adjust fonts, colors, margins, and padding without ever needing to look at HTML or CSS, making this technical, traditionally code-based task more approachable for designers. XRAY promotes the use of design systems by only suggesting styling options that exist in the current design system and highlighting where current aesthetics violate the design system. XRAY also improves designer-developer communication by allowing people in different locations are able to make live edits to a website collaboratively. XRAY allows users to export a document with all changes at the end of their session. Moreover, a 12 person user study with novices and a 12-person user study with professional designers showed that people were xx% more efficient, yy% more successful, and experimented more by using zz% more styles when they used XRAY, than when they used the standard industry tools.

ACM Classification Keywords

H.5.m. Information Interfaces and Presentation (e.g. HCI): Miscellaneous; See http://acm.org/about/class/1998/ for the full list of ACM classifiers. This section is required.

Author Keywords

Web design; design systems; inspector tools; experimentation; human factors; developers; designers.

INTRODUCTION

By definition, designers plan the UI of software and developers write the code. Although some designers can code and some developers are artistically savvy, most professionals in industry specialize. Creating a finished projected is a collaborative and iterative process. For example, if a company was preparing to launch a new website, the designer would create a mockup of how the product should look and send it to the developer. The developer would code it, and send it back to the designer. No first draft is flawless; the designer would next communicate the mistakes to the developer.

CHI'16, May 07–12, 2016, San Jose, CA, USA

ISBN 123-4567-24-567/08/06…$15.00

DOI: http://dx.doi.org/10.475/123_4

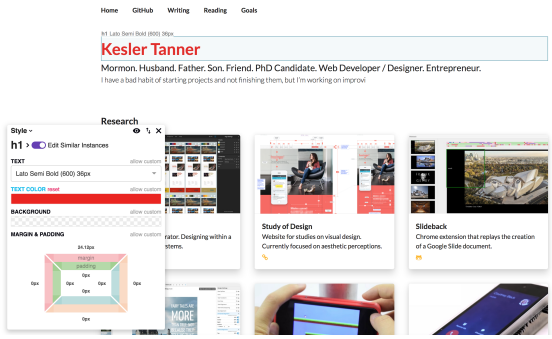

Figure 1. A screenshot of the XRAY user interface and the menu for [fonts? colors? margin/padding?]

Some designers will meet with the developers, sitting next to them and explaining what needs to be changed. However, that is very time-consuming for both the developer and designer. Other designers might send an email with a checklist of what needs to be fixed, but matching the block of test to the website gets confusing very quickly. In reality, it is most common for the developer to "redline" by marking up a screenshot of the website using tools like Sketch or InVision, such as in Figure 2 as taken from Moore's 2017 Medium article [29].

The ability to write comments on a screen simplifies the revision process. Instead of writing an essay for each bullet point ("the icon of a drill on the top left of the home page should be enclosed by a circle instead of a square"), the designer can draw an arrow to the icon they are dissatisfied with and write "should be a circle."

However, when it comes to colors, padding, and other sizing issues, detail is key. Instead of saying "this black should match the black background on the homescreen", why not just tell the developer the hexcode of the color it should be? Additionally, prior work shows the benefits of experimentation, yet writing notes on screenshots does not really provide that opportunity. Ideally, when the designer notices that the line height is too large, they would have the opportunity to see what the page would look like with 1.1em, 1.2em, and 1.3em before telling the developer which one is best.

Currently, the developers are gatekeepers: they control the design and aesthetics of a finished product even though they do not have the knowledge or training that a designer does.

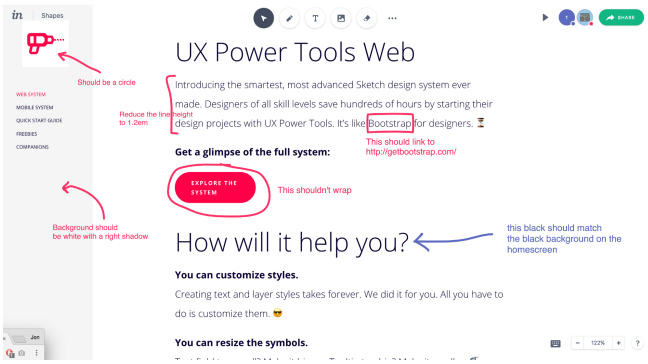

Figure 2. Example of a "redline" or markup of website that a designer might send to a developer

The designers are in the uncomfortable position of knowing exactly how the final product should look, but are unable to create it themselves.

In order to provide designers with more control, many companies are creating design systems. Just as developers use a style guide in order to have a standardized way of doing things, designers create design systems. Most companies strive to be consistent in their use of colors and fonts across websites, emails, advertising campaign, and apps [cite]. Design systems evolved from these style guides. Couldwell walks through how WeWork created a design system which contains not only font families, font sizes, and colors, but also templates, patterns (which include recurring elements or practices like navigation, cards, tables, or alerts), and components (pre-built widgets, such as buttons, sliders, or inputs, selects, toggles, avatars, tooltips, and etc) [6].

When developers and designers face seemingly small questions like "How should we format the time/date/currency?" or "What kind of color/typeface/button should I use?" or "Should this be sentence case or title case?", the design system can evolve to hold these answers [5]. Often a [file type] or [photoshop file], the design system can contain images of what these fonts, colors, and widgets look like, as well as the code to make them. This prevents developers from making the same buttons over and over for each website, and ensures that the color, shadow, and animated reaction for each click or hover over the button will be exactly the same [5].

Although the use of design systems is spreading rapidly in industry, we are not aware of any research regarding the creation or use of them. Companies including [list them] are currently using them; some began creating their design system as early as [year].

In order to 1. improve the experience of web designers, 2. improve their collaboration with developers, and 3. to promote the use of design systems, we built XRAY (Figure 1), design-oriented inspector tools for developers. Unlike traditional inspector tools, XRAY allows the designer to adjust fonts, colors, margins, and padding without ever needing to look at HTML or CSS, making this technical, traditionally code-based task more approachable for designers.

XRAY makes the following contributions:

- A novel set of inspector tools, designed for those who do not need to understand how HTML and CSS work. XRAY allows users to adjust fonts, colors, margins, and padding on live sites without ever looking at any code.
- XRAY focuses on design systems. By only suggesting choices that are in the current design system, it helps novice designers be consistent. XRAY provides the option to override these design guidelines, allowing users to expand the design system by adding new rules. When users redefine an aspect of their design system, XRAY automates consistency, letting the changes percolate through not only the current page but the entire website, persisting even after page reloads. XRAY highlights where a site breaks from the design systems, letting users quickly identify possible issues.
- Designer-developer communication is improved because people in different locations are able to make live edits to a website collaboratively. At the end of a session, designers can export a document with all of their final changes noted in HTML and CSS to give to the developer.
- A within-subject evaluation of XRAY in comparison to the industry standard of using Sketch(c) to note flaws in a website. We had a total of 24 partcipants, half of whom were design professionals with a minimum of five years' experience, and half of whom were design students at our university. Our results showed that people were xx% more efficient, yy% more successful, and experimented more by using zz% more styles when they used XRAY, than when they used the standard industry tools.

RELATED WORKS

Currently, in industry, the best way for designers to give feedback to developers is through marked up screenshots created in Sketch. The purpose of XRAY is to allow designers more control over the design of a website. In essence, XRAY is a designer's version of the developer tools that exist in most browsers. First, we discuss research and tools that were created to assist design. Second, we discuss research and tools for development. Third, we discuss why XRAY is needed to help bridge the divide between developers and designers.

Designers

Examples play a critical role in the creative process [16]. Research has discussed the role of examples in non-routine design [3], how different examples are used in design [16], and the impact of examples [33]. Researchers have created galleries of designs in order to provide designers with examples.

Because web search has historically been done with keywords [11], looking for examples with specific characteristics can be difficult. The 1997 Design Galleries presented users with a variety of options to assist in parameter setting for animation, image rendering, or volume rendering [27]. More recent work has included curated databases of web designs, allowing users to search for examples with similar qualities or look for examples that vary on a particular axis [25], search for examples with stylistic keywords [34] or via the input of a rough sketch [14].

The inherent drawback to human-curated databases is the time required to create and maintain them. Data mining websites, apps and other software for design or other information is discussed in the related works of this thesis [1]. Work has included platforms to collect and annotate web designs on small-scale of 300 web pages [35] to Webzeitgeist, a repository of over 100,000 web pages and 100 million individual design elements [23]. Other work, after mining and extracting such client-side code, has focused on the gathering of Java Script and other interactive aspects of a UI [26]. Other tools for designers automatically reformat websites [36] or text for different aspect ratios as discussed in this literature review [21] or use machine learning to generate alternate page layouts [31]. Other applications involved in mining the web and extracting content include a 2005 paper that determined [??estimated??] 40-50% of the web is template content, or "common content or formatting that appears on multiple pages of a site" [12].

With the rise of companies that sell template-based create-your-own websites (like Wix, WordPress, Shopify, Squarespace, and GoDaddy), it is easy to imagine that number has grown dramatically larger in the past 15 years. The tool Bricolage was created with the goal of allowing any existing website to become a template. It facilitates the transfer of the content of one website into the layout and design of another after learning on human-generated exemplars [24]. However, Bricolage is ML, and gives users no control over which aspects of a design to keep.

Tools like Bricolage that permit users to edit live websites are most similar to XRAY; however, to our knowledge all other tools require some level of proficiency with code and are focused on assisting users to select lines of code to copy into their own projects. The work that we have discussed helps designers get inspired by finding new examples or create entirely new layouts, but is focused on large changes. XRAY moves to fill this gap by facilitating designers' work with fonts, colors, and otherwise performing minor tweaks to designs. Moreover, unlike prior works, XRAY facilitates the use of design systems allowing a site to have an organized system of templates, patterns, and components.

Developers

Just as designers are often inspired by examples, developers often browse the web looking for snippets of code. Recent work has included search engines designed to gather both explanations and code examples onto one results page [19] as well as tools designed to help programmers better understand example code. Micro-explanations, output from the Tutoron browser extension, explain snippets of CSS or unix code as well as regular expressions [15].

In addition to a browser's inspector tools, there is research about alternative software to assist with debugging one's own code, as well as plugins that help users decide what code is needed to get a specific output when inspecting others' code. Unravel is a chrome extension that helps developers track and visualize changes to the HTML, CSS, and JavaScript of a live website [17].

The Rehearse development environment highlights each line of example code as it is executed, and upon the user's selection of a line of relevant code, automatically identities other lines of code that are likely to be pertinent [32]. The Java Whyline posits that it is difficult to determine what code to inspect, and allows developers to work backwards from the code output to determine the cause [22].

FireCrystal is a Firefox extension that helps users identify which lines of HTML, CSS, or JavaScript code cause particular results [30], thus enabling the user to borrow that code for their own projects. WebCrystal was inspired by FireCrystal, but is geared towards novices and allows users to ask questions about recreating particular aspects of an element on a web page [4]. Other work has also focused on novices, studying how they use example code [20] or creating interfaces to help them understand which lines of code are needed to get the desired output [13].

Telescope was designed for users browsing a large breadth of code (like the source code for a large website) to help them find code of interest [18]. Chickenfoot, a programming system embedded in Firefox, was designed to help developers modify the behavior of live websites with no access to the source code [2]. Scotty was designed to alter Mac OS applications at runtime, without access to the source code [10].

These tools help developers better understand live sites and debug their own code, but all require a certain level of coding proficiency. XRAY bridges that gap by creating a new user interface so that designers can have all the control available in inspector tools without needing to write or read any code. As show in figure 1, XRAY could be classified as an web augmentation plugin, as described in Diaz and Arello's 2015 paper: augmented reality for the internet [7].

Previous work, such as WADE [28] and Prefab [9, 8], has simplified the modification of the UI of existing GUIs.

Bridging the gap

Current work often leans on the assumption that designers can code and coders can design. That isn't what reality is like: many people in industry can either code OR design. Although there are many phenomenal tools for designers, most are either geared towards inspiring designers with examples and new ideas, or other big-picture changes, like entirely new layouts. These tools do not give designers sufficient control to assist in the polishing of a final project.

Tools that facilitate the tweaking of a near-finished project are geared towards developers and require at least some coding proficiency. We move to bridge this gap between developers and designers by allowing designers to work in the same medium as the developers. Instead of spending time taking screenshots, and writing directions in plain text, with XRAY designers are able to get the website looking precisely the way it should, and then send developers the code required to implement the changes.

INTERFACE AND FEATURES OF XRAY

Created to assist designers working on websites, XRAY is a chrome plugin written in JavaScript [check that] and is similar

to inspector tools. "WebCrystal accesses the document object model (DOM) of the web page, and treats every DOM node in the web page as a web element." Users can access XRAY by [pressing the extension button in the top right of their screen], as shown in Figure 1. Once XRAY is turned on, users can alter the HTML and CSS of a website of their choice by using buttons and menus, without looking at a single line of code.

Fonts (family, weight, color)

In addition to providing the font families, weights, and colors that are already in the design system, XRAY automatically downloads Google fonts to make them available on the website. This is often a very tedious task that requires the designer to either install the fonts locally or have access to the code.

Whitespace (margin and padding)

Currently, when designers are redlining a document, they only see the whitespace. XRAY shows the padding and margin so that designers can better learn the difference in order to more effectively communicate with developers.

Design system

XRAY preforms a live audit by highlighting aspects of a website that fall outside of the existing design system to make them easier to identify. This gives developers a change to decide if they should expand the design system by including this outlier, or if they should bring the design into alignment by being more consistent and fixing the outlier.

Experimentation

In order to increase experimentation, XRAY lets users copy and paste styles from one element to the next. These changes do not need to be from the same webpage. Users can toggle on and off aspects of those changes, and aspects from several sources can be pasted into the same element. XRAY automates consistency, letting the changes percolate through not only the current page but the entire website, persisting even after page reloads. Users get an instant preview of those changes, which allows them to experiment more because they no longer need to adjust each different heading to see how it would look.

Collaboration

Two people can make live edits to a website collaboratively. While our users were mixed about the practicality of this feature, one benefit is that designers can share the session with a developer so they can have a live editor with the changes, instead of a static list of the differences.

Download/explore final changes

Instead of a series of redlined screenshots, users can download the final visual changelog from XRAY to give to the developer. Instead of descriptions of requested changes, the changelog is code, making it easy for the developer to make the website look exacly as it did when the designer finished tweaking and adjusting it using XRAY. This will eliminate the need of further back-and-forth communications.

USER STUDY

Participants

12 students 12 professionals

Tasks

A, B, C (or) A, B

Procedure

And fill in this section too

Results

And put some graphs and stats and stuff in here

CONCLUSION/DISCUSSION

Discuss benefits...

- promotes use of design system
- improves designer-developer communication
- this gets designers into the same medium that developers are using (ie, "writing code" except they are still in their comfort zone")
- keeps the designers in their element: they can ignore the code entirely
- lets the developers get back in their element: designers are giving them code to implement instead of vague directions

Discuss harm/risks/dangers/drawbacks...

- ...I need to think about this...

Discuss limitations of our work

- margin/padding issue is not yet solved
- there needs to be more conversation about design systems
- does not support absolute positioning (cannot grab a piece of the page and move it to a different part of the web page)
- we didn't actually implement, but COULD implement some kind of live/real-time collaborative editing thing
- talk about whether or not we are really facilitating experimentation

Discuss future work/room to grow...

- see the limitations and drawback sections above.

REFERENCES

1. Khalid Ahmed Alharbi and Tom Yeh. 2016. *A Deep and Longitudinal Approach to Mining Mobile Applications*. Ph.D. Dissertation.
2. Michael Bolin, Matthew Webber, Philip Rha, Tom Wilson, and Robert C. Miller. 2005. Automation and Customization of Rendered Web Pages. In *Proceedings of the 18th Annual ACM Symposium on User Interface Software and Technology (UIST '05)*. ACM, New York, NY, USA, 163–172. DOI: http://dx.doi.org/10.1145/1095034.1095062
3. Nathalie Bonnardel. 1999. Creativity in Design Activities: The Role of Analogies in a Constrained Cognitive Environment. In *Proceedings of the 3rd Conference on Creativity & Cognition (C&C '99)*. ACM, New York, NY, USA, 158–165. DOI: http://dx.doi.org/10.1145/317561.317589

4. Kerry Shih-Ping Chang and Brad A. Myers. 2012. WebCrystal: Understanding and Reusing Examples in Web Authoring. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '12)*. ACM, New York, NY, USA, 3205–3214. DOI: http://dx.doi.org/10.1145/2207676.2208740

5. Andrew Couldwell. 2017a. Laying the foundations for system design: Great digital products are built upon solid foundations. *Medium* (Dec 2017). https://medium.com/owl-studios/laying-the-foundations-7e503ef2120f

6. Andrew Couldwell. 2017b. Plasma design system: Creating and documenting a product design system. *Medium* (Jan 2017). https://medium.com/owl-studios/plasma-design-system-4d63fb6c1afc

7. Oscar Díaz and Cristóbal Arellano. 2015. The Augmented Web: Rationales, Opportunities, and Challenges on Browser-Side Transcoding. *ACM Trans. Web* 9, 2, Article 8 (May 2015), 30 pages. DOI: http://dx.doi.org/10.1145/2735633

8. Morgan Dixon and James Fogarty. 2010. Prefab: Implementing Advanced Behaviors Using Pixel-based Reverse Engineering of Interface Structure. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '10)*. ACM, New York, NY, USA, 1525–1534. DOI: http://dx.doi.org/10.1145/1753326.1753554

9. Morgan Dixon, Daniel Leventhal, and James Fogarty. 2011. Content and Hierarchy in Pixel-based Methods for Reverse Engineering Interface Structure. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '11)*. ACM, New York, NY, USA, 969–978. DOI: http://dx.doi.org/10.1145/1978942.1979086

10. James R. Eagan, Michel Beaudouin-Lafon, and Wendy E. Mackay. 2011. Cracking the Cocoa Nut: User Interface Programming at Runtime. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology (UIST '11)*. ACM, New York, NY, USA, 225–234. DOI: http://dx.doi.org/10.1145/2047196.2047226

11. James Fogarty, Desney Tan, Ashish Kapoor, and Simon Winder. 2008. CueFlik: Interactive Concept Learning in Image Search. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '08)*. ACM, New York, NY, USA, 29–38. DOI: http://dx.doi.org/10.1145/1357054.1357061

12. David Gibson, Kunal Punera, and Andrew Tomkins. 2005. The Volume and Evolution of Web Page Templates. In *Special Interest Tracks and Posters of the 14th International Conference on World Wide Web (WWW '05)*. ACM, New York, NY, USA, 830–839. DOI: http://dx.doi.org/10.1145/1062745.1062763

13. Paul Gross, Jennifer Yang, and Caitlin Kelleher. 2011. Dinah: An Interface to Assist Non-programmers with Selecting Program Code Causing Graphical Output. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '11)*. ACM, New York, NY, USA, 3397–3400. DOI: http://dx.doi.org/10.1145/1978942.1979448

14. Yasunari Hashimoto and Takeo Igarashi. 2005. Retrieving Web Page Layouts using Sketches to Support Example-based Web Design. In *Eurographics Workshop on Sketch-Based Interfaces and Modeling*, Joaquim Armando Pires Jorge and Takeo Igarashi (Eds.). The Eurographics Association. DOI: http://dx.doi.org/10.2312/SBM/SBM05/155-164

15. Andrew Head and Marti A. Hearst. Tutorons: Generating Context-Relevant, On-Demand Explanations and Demonstrations of Online Code. (????).

16. Scarlett R. Herring, Chia-Chen Chang, Jesse Krantzler, and Brian P. Bailey. 2009. Getting Inspired!: Understanding How and Why Examples Are Used in Creative Design Practice. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '09)*. ACM, New York, NY, USA, 87–96. DOI: http://dx.doi.org/10.1145/1518701.1518717

17. Joshua Hibschman and Haoqi Zhang. 2015. Unravel: Rapid Web Application Reverse Engineering via Interaction Recording, Source Tracing, and Library Detection. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology (UIST '15)*. ACM, New York, NY, USA, 270–279. DOI: http://dx.doi.org/10.1145/2807442.2807468

18. Joshua Hibschman and Haoqi Zhang. 2016. Telescope: Fine-Tuned Discovery of Interactive Web UI Feature Implementation. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology (UIST '16)*. ACM, New York, NY, USA, 233–245. DOI: http://dx.doi.org/10.1145/2984511.2984570

19. Raphael Hoffmann, James Fogarty, and Daniel S. Weld. 2007. Assieme: Finding and Leveraging Implicit References in a Web Search Interface for Programmers. In *Proceedings of the 20th Annual ACM Symposium on User Interface Software and Technology (UIST '07)*. ACM, New York, NY, USA, 13–22. DOI: http://dx.doi.org/10.1145/1294211.1294216

20. M. Ichinco and C. Kelleher. 2015. Exploring novice programmer example use. In *2015 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. 63–71. DOI: http://dx.doi.org/10.1109/VLHCC.2015.7357199

21. Charles Jacobs, Wilmot Li, Evan Schrier, David Bargeron, and David Salesin. 2003. Adaptive Grid-based Document Layout. *ACM Trans. Graph.* 22, 3 (July 2003), 838–847. DOI:http://dx.doi.org/10.1145/882262.882353

22. Andrew J. Ko and Brad A. Myers. 2009. Finding Causes of Program Output with the Java Whyline. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '09)*. ACM, New York, NY, USA, 1569–1578. DOI: http://dx.doi.org/10.1145/1518701.1518942

23. Ranjitha Kumar, Arvind Satyanarayan, Cesar Torres, Maxine Lim, Salman Ahmad, Scott R. Klemmer, and Jerry O. Talton. 2013. Webzeitgeist: Design Mining the Web. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '13)*. ACM, New York, NY, USA, 3083–3092. DOI: http://dx.doi.org/10.1145/2470654.2466420

24. Ranjitha Kumar, Jerry O. Talton, Salman Ahmad, and Scott R. Klemmer. 2011. Bricolage: Example-based Retargeting for Web Design. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '11)*. ACM, New York, NY, USA, 2197–2206. DOI: http://dx.doi.org/10.1145/1978942.1979262

25. Brian Lee, Savil Srivastava, Ranjitha Kumar, Ronen Brafman, and Scott R. Klemmer. 2010. Designing with Interactive Example Galleries. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '10)*. ACM, New York, NY, USA, 2257–2266. DOI: http://dx.doi.org/10.1145/1753326.1753667

26. Josip Maras, Jan Carlson, and Ivica Crnkovi. 2012. Extracting Client-side Web Application Code. In *Proceedings of the 21st International Conference on World Wide Web (WWW '12)*. ACM, New York, NY, USA, 819–828. DOI: http://dx.doi.org/10.1145/2187836.2187947

27. J. Marks, B. Andalman, P. A. Beardsley, W. Freeman, S. Gibson, J. Hodgins, T. Kang, B. Mirtich, H. Pfister, W. Ruml, K. Ryall, J. Seims, and S. Shieber. 1997. Design Galleries: A General Approach to Setting Parameters for Computer Graphics and Animation. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '97)*. ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 389–400. DOI: http://dx.doi.org/10.1145/258734.258887

28. Xiaojun Meng, Shengdong Zhao, Yongfeng Huang, Zhongyuan Zhang, James Eagan, and Ramanathan Subramanian. 2014. WADE: Simplified GUI add-on development for third-party software. (04 2014).

29. Jon Moore. 2017. InVision Freehand: A 21st Century Case for Digital Whiteboards. *Medium* (Aug 2017). https://medium.com/ux-power-tools/invision-freehand-digital-whiteboarding-bf83639c1184

30. S. Oney and B. Myers. 2009. FireCrystal: Understanding interactive behaviors in dynamic web pages. In *2009 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. 105–108. DOI: http://dx.doi.org/10.1109/VLHCC.2009.5295287

31. P. OâĂŹDonovan, A. Agarwala, and A. Hertzmann. 2014. Learning Layouts for Single-PageGraphic Designs. *IEEE Transactions on Visualization and Computer Graphics* 20, 8 (Aug 2014), 1200–1213. DOI: http://dx.doi.org/10.1109/TVCG.2014.48

32. Joel Branndt Vignan Pattamatta, William Choi, Ben Hsieh, and Scott R. Klemmer. Rehearse: Helping Programmers Adapt Examples by Visualizing Execution and Highlighting Related Code. (????).

33. A. Terry Purcell and John S. Gero. 1992. Effects of examples on the results of a design activity. *Knowl.-Based Syst.* 5 (1992), 82–91.

34. Daniel Ritchie, Ankita Arvind Kejriwal, and Scott R. Klemmer. 2011. D.Tour: Style-based Exploration of Design Example Galleries. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology (UIST '11)*. ACM, New York, NY, USA, 165–174. DOI: http://dx.doi.org/10.1145/2047196.2047216

35. Arvind Satyanarayan, Maxine Lim, and Scott Klemmer. 2012. A Platform for Large-scale Machine Learning on Web Design. In *CHI '12 Extended Abstracts on Human Factors in Computing Systems (CHI EA '12)*. ACM, New York, NY, USA, 1697–1702. DOI: http://dx.doi.org/10.1145/2212776.2223695

36. Nishant Sinha and Rezwana Karim. 2013. Compiling Mockups to Flexible UIs. In *Proceedings of the 2013 9th Joint Meeting on Foundations of Software Engineering (ESEC/FSE 2013)*. ACM, New York, NY, USA, 312–322. DOI:http://dx.doi.org/10.1145/2491411.2491427