Designing User Interfaces for Studying Apprenticeship Learning at Scale

Veronica Rivera
March 17, 2021

Introduction and Background

MOOCs have allowed for greater dissemination of educational content, and have been an important part of expanding access to learning resources, especially in Computer Science (CS) education. However, MOOCs do not adequately equip students with the skills needed to perform expert-level work, which are best learned through experiential and apprenticeship learning [2]. In CS education specifically, this kind of learning is often obtained through real-world learning experiences such as summer internships, undergraduate research opportunities, and involvement in capstone projects.

However, current experiential and apprenticeship learning opportunities in CS can be difficult to access. For instance internships and undergraduate research opportunities require applications and interviews, which means they are competitive from the start. Given the myriad of learning benefits that hands-learning opportunities provide for individuals wanting to learn CS, it is important to consider how these opportunities could reach more people.

Scaling experiential and apprenticeship learning is difficult because they require a significant time commitment from mentors who oversee students, guide them through tasks, and provide instructional scaffolding. This led us to ask: *How might we design online learning environments that support apprenticeship learning at scale? How should we design instructional support and materials within these platforms to enable learning without the need for mentors?* Currently we are planning a study to address these questions using Causeway, an online platform that enables individuals to learn web development by following a series of learning pathways (micro-roles) as they work on real-world projects [4]. Two learning interfaces were designed and created for this study: Classroom and Studio. Both platforms contain the same learning materials and assignments. However, they differ in how the materials are structured and presented to students.

Design Process and Iteration

I joined the project in Summer 2019 when the initial design for the Classroom interface was in the final stages of being developed by other members of the Tech4Good Lab. My task from that point onwards has been to conduct usability tests of the two interfaces and make design decisions for future iterations so that the interfaces can be used in our study. I have led several teams of undergraduate students in doing this work throughout multiple quarters (Summer, Fall 2019; Winter, Spring, Fall 2020; Winter 2021).

Prior to the COVID-19 pandemic we conducted usability tests by inviting UCSC undergraduate students to the lab and having them use the relevant Causeway interface for a few hours. During this time everyone on the team would take notes and the participant would talk aloud about their experience. We also interviewed students for about 15 minutes to dig deeper into their experiences with the platform. At the end I would lead discussions with my team about our observations and write up design recommendations for future iterations. Starting in March 2020 we followed the same process, but began conducting virtual usability tests over Zoom while having the participant share their screen as they work through the platform.

Throughout this process we have focused on how to design the interfaces in a way that best presents learning resources and information to users to help them complete the projects within Causeway successfully and want to continue using the platform. *In this section I will describe how we used participant observations, usability tests, the think-aloud method, and participant interviews to test the interfaces with UCSC undergraduate students, and the design iterations that arose from that process.*

Classroom Interface

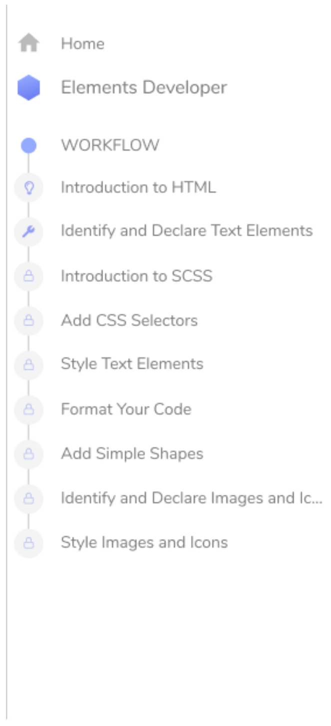
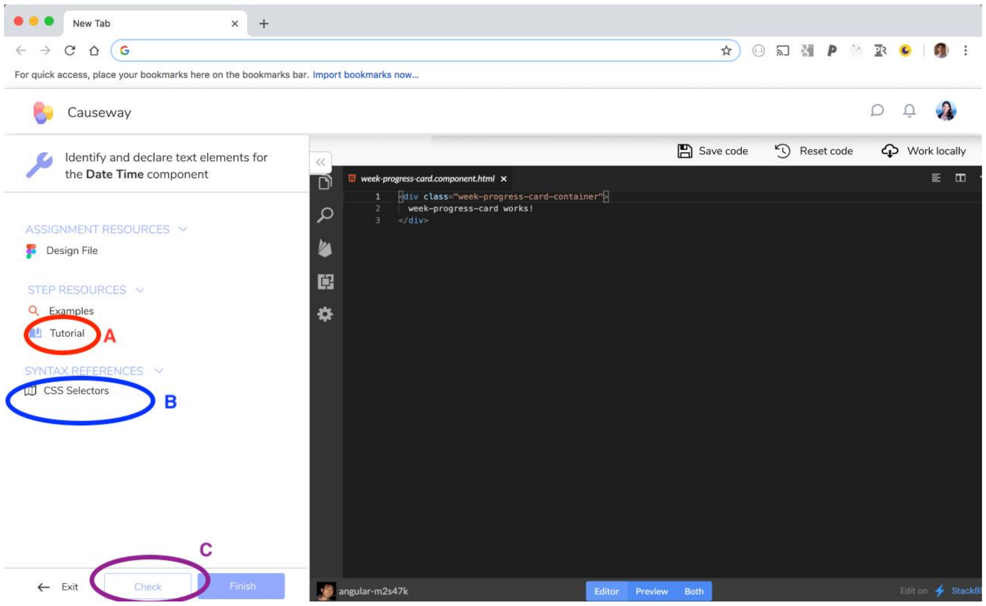


Figure 1. Sample Classroom workflow for the Elements role

In Classroom, projects are broken up into several roles such as “Elements”, where students are responsible for identifying, implementing, and styling simple elements, and “Layout” where they proceed to use flexbox-based positioning to style the elements from the previous role. Within each role the users are given a step-by-step workflow to follow as they learn and work on projects (Fig. 1). One of the key components of traditional apprenticeship learning is having an expert mentor provide scaffolding to novices [1,3]. Our goal with creating this workflow was to model this type of scaffolding and guidance in Causeway without the need for a mentor. Within the main workspace (Fig. 2), students are provided with a set of written tutorial guides (Fig. 2(A)). These guides provide examples that model concepts and how to put them into practice, similar to the modeling provided by mentors in traditional apprenticeship learning [1,3]. To support the guides we added a section called Syntax references to the left side panel (Fig. 2(B)). The syntax references provide students with a list of relevant tags and selectors that are introduced in the guides for that role. Finally, students are given a sample solution they can use to check their answer (Fig. 2(C)). This was added to further enable students to work independently without the need for expert guidance.

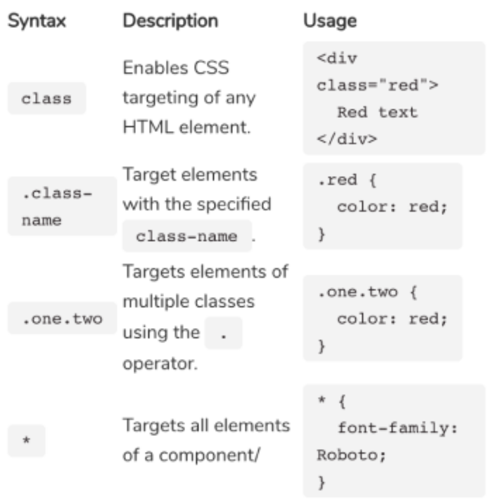
We noticed in our early usability tests that *it was not obvious to students how to navigate the main workspace* (Fig. 2) even after it was introduced to them previously in Causeway's onboarding section. Students would enter the workspace for the first time and not know what they needed to do first. In some of our participant interviews students told us that by the time they finished onboarding they had already forgotten how to use the workspace. To remedy this issue, we decided to add a walkthrough video to the assignment resources dropdown in later iterations that briefly explains how to use the workspace. We considered just including a way for students to go back to the onboarding section that explains this, but that would have required them to scroll through multiple pages of information before finding what they needed. The video presents a much faster and straightforward way to remind students how to navigate the workspace and get started. In future usability tests we noticed that students really liked the video and used it often.



```
<div class="week-progress-card-container">
  week-progress-card works!
</div>
```

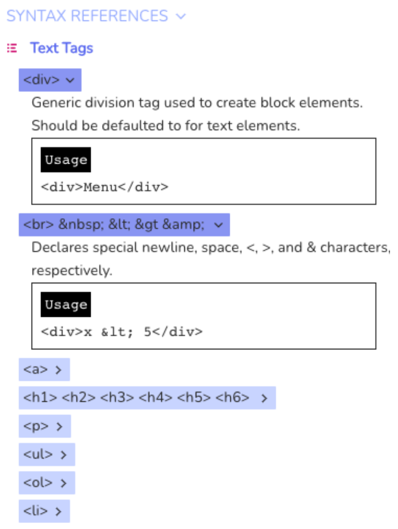
Figure 2. Early Classroom interface

Another important issue we found was that students found the guides difficult to read. The guides consist of a lot of text, and *several students expressed that they do not like reading and instead relied heavily on the syntax references drop down*. Oftentimes we observed that students were using the syntax references more than the guides themselves. Therefore, we made the syntax references dropdown easier to use. In early versions of Classroom several tags and a brief description of their usage were listed side by side in columns (Fig. 3A). Many of the descriptions for different tags would run into each other making it difficult to discern the individual tags. The first thing we tried was to list all the tags and display their descriptions on hover. However, this did not allow students to compare different tags and their usage, which is important in helping novices learn. In later design iterations we made each individual tag a drop down that students could click to open a longer description of the usage (Fig. 3B)



Syntax	Description	Usage
`class`	Enables CSS targeting of any HTML element.	`<div class="red"> Red text </div>`
`.class-name`	Target elements with the specified `class-name` .	`.red { color: red; }`
`.one.two`	Targets elements of multiple classes using the `.` operator.	`.one.two { color: red; }`
`*`	Targets all elements of a component/	`* { font-family: Roboto; }`

Figure 3A. Syntax references initial



Generic division tag used to create block elements. Should be defaulted to for text elements.

```
Usage
<div>Menu</div>
```

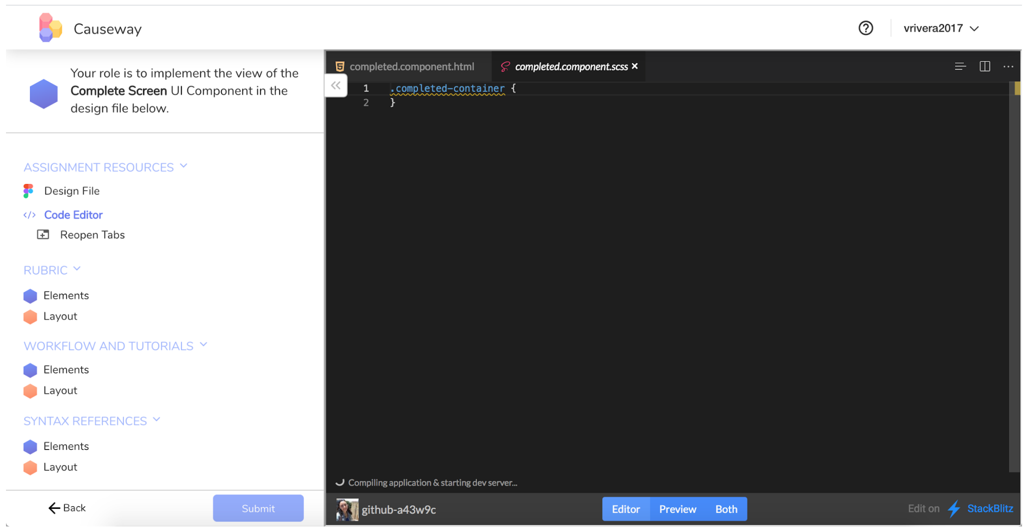
Declares special newline, space, <, >, and & characters, respectively.

```
Usage
<div>x &lt; 5</div>
```

Figure 3B. Syntax references final

Probably one of the most interesting observations was that *the step-by-step presentation of the guides and workflow did not match many students' mental models*. Many students wanted to go further in a single step than the guides would allow. We even had one student that was doing his own thing completely rather than working on the assigned project. During the participant interview with this student, we realized he had a completely different understanding of Causeway. He thought that Causeway provided the tools to support his creativity as he worked on his own projects freely. He even commented that he enjoyed this more than platforms like Coursera and Codecademy that he's used in the past because those platforms bind students to a strict workflow. Seeing how some students wanted to go further in a single step than is allowed by the Classroom interface, we created a new interface, Studio. Studio provides students with more freedom to incorporate the resources in whatever order they want.

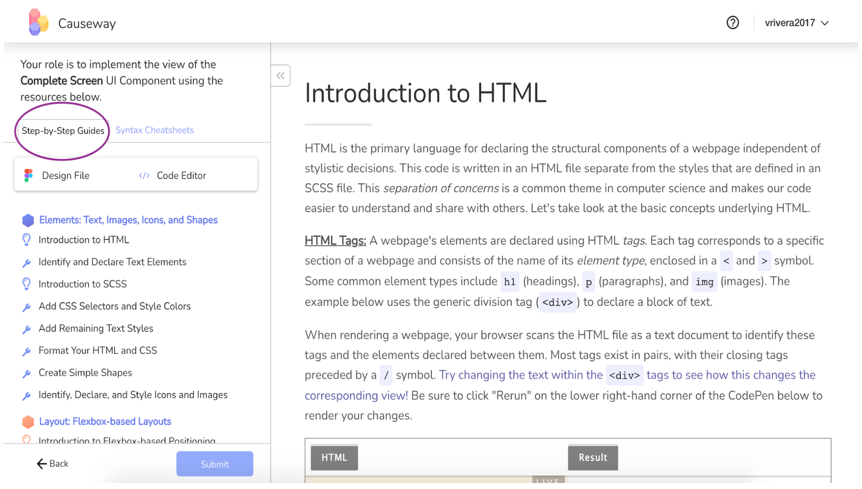
Studio Interface


Your role is to implement the view of the **Complete Screen** UI Component in the design file below.

```
.completed-container {
}
```

Figure 4. Early Studio Interface

In Studio, students are assigned a project and given many of the same resources (e.g. guides and syntax references) as in Classroom, but they are not constrained to a predetermined workflow. Additionally, students are not given sample solutions. The decision to remove the workflow and solutions were prompted by our previous observations of students needing more flexibility in Classroom. By having both Classroom and Studio on opposite ends of the spectrum, we might be able to make observations about how much scaffolding is appropriate when we conduct our study.


Your role is to implement the view of the **Complete Screen** UI Component using the resources below.

Introduction to HTML

HTML is the primary language for declaring the structural components of a webpage independent of stylistic decisions. This code is written in an HTML file separate from the styles that are defined in an SCSS file. This *separation of concerns* is a common theme in computer science and makes our code easier to understand and share with others. Let's take look at the basic concepts underlying HTML.

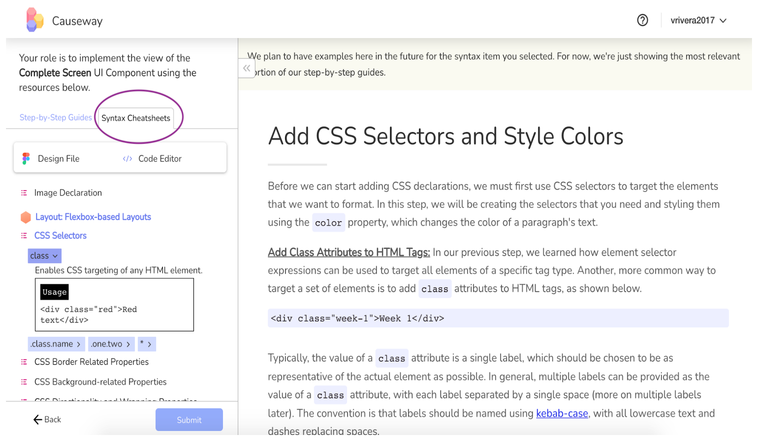
HTML Tags: A webpage's elements are declared using HTML *tags*. Each tag corresponds to a specific section of a webpage and consists of the name of its *element type*, enclosed in a `<` and `>` symbol. Some common element types include `h1` (headings), `p` (paragraphs), and `img` (images). The example below uses the generic division tag (`<div>`) to declare a block of text.

When rendering a webpage, your browser scans the HTML file as a text document to identify these tags and the elements declared between them. Most tags exist in pairs, with their closing tags preceded by a `/` symbol. Try changing the text within the `<div>` tags to see how this changes the corresponding view! Be sure to click "Rerun" on the lower right-hand corner of the CodePen below to render your changes.

Causeway
vrivera2017
Step-by-Step Guides
Syntax Cheatsheets
Design File
Code Editor
Elements: Text, Images, Icons, and Shapes
Introduction to HTML
Identify and Declare Text Elements
Introduction to SCSS
Add CSS Selectors and Style Colors
Add Remaining Text Styles
Format Your HTML and CSS
Create Simple Shapes
Identify, Declare, and Style Icons and Images
Layout: Flexbox-based Layouts
Introduction to Flexbox-based Positioning
Back
Submit
HTML
Result

Figure 5A. Guides organized by workflow

In early designs of Studio we presented the guides and syntax references organized by Classroom roles, such as “Elements” and “Layout” (Fig. 4). In doing this, we were essentially presenting the guides following the same Classroom workflow, without forcing students to follow the workflow. It was just a way for the guides to be organized and give students some guidance in case they needed a place to start. However, in early usability tests we noticed that *students still chose to use the syntax references rather than the guides.* Some students didn't even open the guides at all, especially those who already had some experience with programming prior to the usability test. One of our goals with Causeway is for students to be able to learn and transfer their knowledge outside the platform, so we were concerned that they might not learn as much by simply using the syntax references instead of reading through the guides. Therefore, we decided to organize the guides in two ways: one using the Classroom step-by-step workflow, and another using the Syntax references (Fig. 5A, 5B). This way, students who would rather use the Syntax references can do so while accessing the guide where those concepts are introduced.



Your role is to implement the view of the **Complete Screen** UI Component using the resources below.

Enables CSS targeting of any HTML element.

```
Usage
<div class="red">Red
text</div>
```

We plan to have examples here in the future for the syntax item you selected. For now, we're just showing the most relevant ɔrtion of our step-by-step guides.

Add CSS Selectors and Style Colors

Before we can start adding CSS declarations, we must first use CSS selectors to target the elements that we want to format. In this step, we will be creating the selectors that you need and styling them using the `color` property, which changes the color of a paragraph's text.

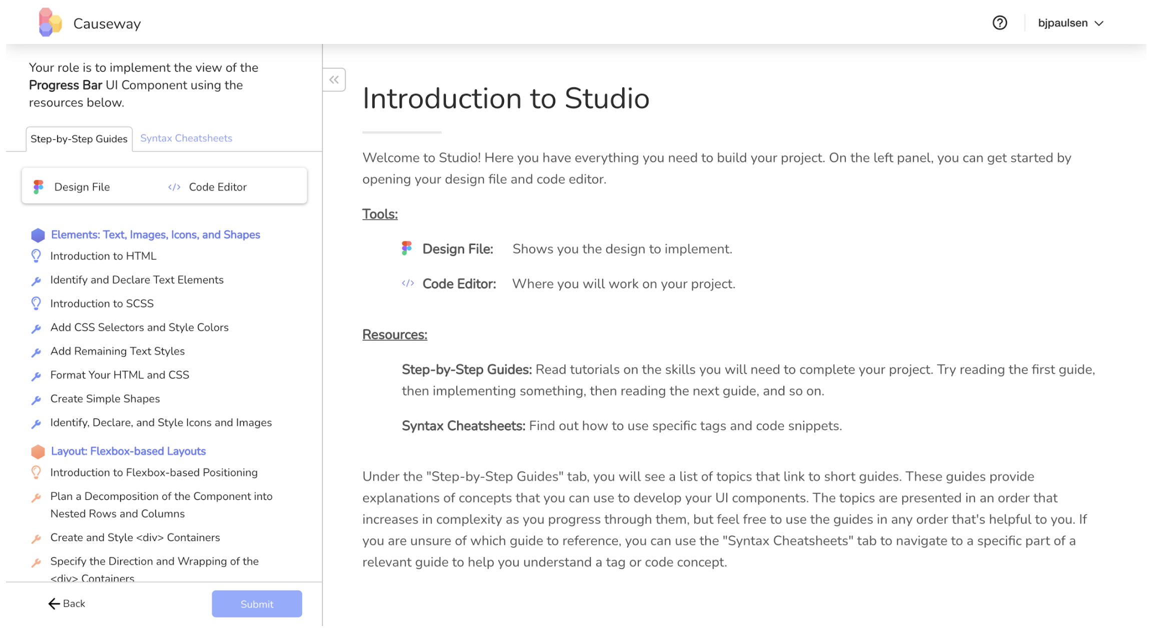
Add Class Attributes to HTML Tags: In our previous step, we learned how element selector expressions can be used to target all elements of a specific tag type. Another, more common way to target a set of elements is to add `class` attributes to HTML tags, as shown below.

```
<div class="week-1">Week 1</div>
```

Typically, the value of a `class` attribute is a single label, which should be chosen to be as representative of the actual element as possible. In general, multiple labels can be provided as the value of a `class` attribute, with each label separated by a single space (more on multiple labels later). The convention is that labels should be named using kebab-case, with all lowercase text and dashes replacing spaces.

Figure 5B. Guides organized by syntax references

We conducted multiple usability tests of the interface in Fig. 5A and 5B but found that *many students did not realize that there are two ways of displaying the guides.* Upon entering the main interface, students were very confused and overwhelmed. Therefore, we decided to add a short introduction upon entering the interface that describes the resources available so that students have something to reference (Fig. 6). We conducted a usability test with a single user and saw that the introduction helped clear up a lot of the confusion we had seen in prior usability tests.



Your role is to implement the view of the **Progress Bar** UI Component using the resources below.

Elements: Text, Images, Icons, and Shapes
- Introduction to HTML
- Identify and Declare Text Elements
- Introduction to SCSS
- Add CSS Selectors and Style Colors
- Add Remaining Text Styles
- Format Your HTML and CSS
- Create Simple Shapes
- Identify, Declare, and Style Icons and Images

Layout: Flexbox-based Layouts
- Introduction to Flexbox-based Positioning
- Plan a Decomposition of the Component into Nested Rows and Columns
- Create and Style <div> Containers
- Specify the Direction and Wrapping of the <div> Containers

Introduction to Studio

Welcome to Studio! Here you have everything you need to build your project. On the left panel, you can get started by opening your design file and code editor.

Tools:

Design File: Shows you the design to implement.

Code Editor: Where you will work on your project.

Resources:

Step-by-Step Guides: Read tutorials on the skills you will need to complete your project. Try reading the first guide, then implementing something, then reading the next guide, and so on.

Syntax Cheatsheets: Find out how to use specific tags and code snippets.

Under the "Step-by-Step Guides" tab, you will see a list of topics that link to short guides. These guides provide explanations of concepts that you can use to develop your UI components. The topics are presented in an order that increases in complexity as you progress through them, but feel free to use the guides in any order that's helpful to you. If you are unsure of which guide to reference, you can use the "Syntax Cheatsheets" tab to navigate to a specific part of a relevant guide to help you understand a tag or code concept.

Figure 6. Current Studio Interface with introduction

Conclusion

In this statement I described the process we used to create two online learning interfaces that model key elements of apprenticeship learning: Classroom and Studio. Through user testing of Classroom we noted three important observations that we addressed in later iterations. First, *it was not obvious to students how to navigate the main workspace.* Second, *students did not like reading large amounts of text and instead relied heavily on the syntax references drop down to learn key concepts.* Third, *the step-by-step presentation of the guides and workflow did not match many students' mental models.* This last observation inspired the design of the Studio interface, where students are given more control over their learning process. In early usability tests of Studio we found that *students chose to use the syntax references rather than the guides,* prompting us to add two ways to organize the guides. Further usability testing showed that *many students did not realize that there are two ways of displaying the guides,* so we added a short introduction describing the resources available upon entering the interface. Our next step is to use these two interfaces in a study to evaluate which features are most helpful to students' learning over a longer period of time.

References

[1] Allan Collins, John Seely Brown, and Ann Holum. 1991. Cognitive apprenticeship: Making thinking visible. *American educator* 15, 3 (1991), 6–11.

[2] Allan Collins, John Seely Brown, and Susan E. Newman. 1988. Cognitive Apprenticeship: Teaching the Craft of Reading, Writing and Mathematics. *Thinking: The Journal of Philosophy for Children* 8, 1 (March 1988), 2–10.

[3] Aziz Ghefaili. 2003. Cognitive apprenticeship, technology, and the contextualization of learning environments. *Journal of Educational Computing, Design & Online Learning* 4, 1 (2003), 1–27.

[4] David T. Lee, Emily S. Hamedian, Greg Wolff, and Amy Liu. 2019. Causeway: Scaling Situated Learning with Micro-Role Hierarchies. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems* (CHI ’19), Association for Computing Machinery, New York, NY, USA, 1–12.