# Blending Accessibility in UI Framework Documentation to Build Awareness

MAULISHREE PANDEY*, Google, USA
TAO DONG, Google, USA

The lack of accessibility awareness among industry professionals is one of the reasons for rampant inaccessible websites and applications. This problem is exacerbated by the industry norm of having a single place dedicated to accessibility in the documentation of UI frameworks, which makes accessibility difficult for developers to discover and implement as part of their workflows. This paper presents the Blended Approach (BA), a novel approach and framework for improving accessibility awareness through documentation. Unlike the conventional practice, it recommends sprinkling and repeating short snippets on accessibility throughout the documentation while linking developers to detailed explanations on the dedicated accessibility page. Thus, BA places the topic of accessibility on an equal footing as other common programming concerns such as performance, security, and UX. As a case study, we applied BA to the onboarding tutorial of Flutter, a popular UI toolkit. The positive feedback we received in our evaluation with 11 professional developers suggests BA can be a viable and effective approach.

CCS Concepts: • **Human-centered computing → User studies**; **Accessibility design and evaluation methods**; • **Software and its engineering → Development frameworks and environments**.

Additional Key Words and Phrases: software developers, accessibility, programming, documentation, UI development

**ACM Reference Format:**
Maulishree Pandey and Tao Dong. 2023. Blending Accessibility in UI Framework Documentation to Build Awareness. In *The 25th International ACM SIGACCESS Conference on Computers and Accessibility (ASSETS '23), October 22--25, 2023, New York, NY, USA.* ACM, New York, NY, USA, 20 pages. https://doi.org/10.1145/3597638.3608380

## 1 INTRODUCTION

In recent years, there has been a growing focus on educating developers about accessibility. Researchers and faculty members are making efforts to teach accessibility as part of computer science curriculums [20, 31, 43]. These efforts aim to make the next generation of software developers more informed about accessibility. However, there is little consideration toward building awareness among developers already employed in the profession and self-taught developers who are not exposed to courses on accessibility. The lack of awareness among industry professionals is one of the reasons for rampant inaccessible websites and applications [35, 36]. Another factor is the growing popularity of cross-platform UI frameworks and applications [7]. Frameworks such as React Native [24], Flutter [12], and Cordova [40] enable developers to target multiple operating systems and devices from a single codebase. However, developers using these frameworks are often unaware of the inconsistent behaviors of resulting applications on assistive technologies. For instance, Pandey *et al.* showed that applications produced by cross-platform frameworks differ across screen readers [26]. The research reported that sighted developers assume the consistency in visual form and functionality, which they normally test and debug, translate to screen readers. Furthermore, they are unaware of how to write code accessibly as part of their development workflows unless educated by their visually impaired developer colleagues.

*The author is a doctoral student at the University of Michigan at the time of writing. The research was done when she was an intern at Google.


Manuscript submitted to ACM

To improve accessibility, we must target awareness-building efforts on developers more broadly. In this paper, we present the Blended Approach (BA) to documenting accessibility, a novel approach and framework for improving accessibility awareness through developer documentation. BA is a departure from the widespread industry norm of having a single place dedicated to accessibility in the documentation of UI frameworks and libraries [30]. Instead, it recommends sprinkling and repeating short snippets on accessibility throughout the documentation while linking developers to detailed explanations on the dedicated accessibility page. Thus, BA places the topic of accessibility at an equal footing as other common programming concerns such as performance, security, and UX.

We utilized the user-centered design process to develop and evaluate BA (see §3). We started by conducting formative interviews with professional UI developers and held conversations with accessibility experts and developers who have worked on accessibility features of UI frameworks. Our goal was to identify what kind of accessibility information is most relevant to UI developers. Informed by the interviews and consultations, we developed six ideas for building accessibility awareness. Next, we organized a design workshop to refine and prepare our ideas for implementation. We zeroed in on the idea of creating short and focused accessibility content that can be integrated into high-traffic pages of a UI framework's official documentation.

We chose Flutter, a popular and open source UI toolkit, for implementation and evaluation of our design idea, which led to the development of our documentation approach. We developed short pieces of accessibility content and added these to a copy of Flutter's official onboarding tutorial. We hosted the modified website for an evaluation study with 11 professional front-end/full-stack developers. As part of the study, we first observed their unprompted and unprimed response to the blended accessibility content, followed by a short interview where we collected specific feedback on the changes we had made. Majority of the participants reacted positively upon seeing instances of accessibility in the tutorial and shared that blended content can help them discover and apply accessibility information more easily.

In summary, we contribute the following:

- Blended approach (BA), a novel approach and framework for documenting accessibility in UI frameworks and libraries. The framework outlines considerations for documentation, categories of accessibility information, and how to represent content within each category through text, video, etc. (see §7.1)
- An end-to-end example of BA's application in Flutter's onboarding tutorial [1], which serves as an example for other UI frameworks and libraries (see §4)
- An evaluation of our implementation, which validates BA and demonstrates developers' readiness for learning about accessibility as part of their development workflows. We also confirm findings reported in prior empirical studies that have investigated accessibility awareness among technology professionals. (see §6)

## 2 RELATED WORK

We first present the benefits and limitations of research efforts that have focused on teaching accessibility to computer science (CS) students. Then we discuss empirical research that has investigated accessibility awareness among developers in the industry.

[1]The Flutter team integrated the accessibility content into its official tutorial in November, 2022: http://web.archive.org/web/20221211163840/https://docs.flutter.dev/get-started/codelab.
Parts of the content were migrated to the new tutorial, available here: https://codelabs.developers.google.com/codelabs/flutter-codelab-first#4

### 2.1 Teaching Accessibility

In the last couple of decades, there have been dedicated efforts to teach accessibility to CS students. Consider Teach Access, a non-profit organization that collaborates with universities, companies, and disability advocacy organizations to impart accessibility education to students in the fields of design, programming, and tech-adjacent university programs [2]. Their mission is to ensure that developers and designers entering the workforce are equipped with accessibility knowledge and apply an inclusion-first approach to their industry projects. For instance, supported by Teach Access, Kearney-Volpe *et al.* modified eighteen different computing and non-computing courses at various universities and programs to cover a range of accessibility topics. They found that students benefited the most from videos, screen reader previews, and in-class discussions [16]. Similarly, researchers have argued that UI and web development courses can be used to teach accessibility guidelines and principles of inclusive guidelines [8, 19]. A few of these courses have required students to collaborate with people with disabilities to ensure a deep understanding of accessibility guidelines [6, 23].

Others have proposed integrating accessibility across all four years of undergraduate CS coursework [43]. To achieve systemic and thoughtful integration, Ko and Ladner suggest that instructors consider modifying a single lecture, followed by adding a lecture, and ultimately adding a course on accessibility [17]. A complementary strategy is to rethink the examples, historical contexts, and motivational problems that CS courses rely on and modify each of these to introduce accessibility content [15].

However, as part of their educational and advocacy efforts, researchers have uncovered challenges in teaching accessibility. Despite the growing emphasis, Teach Access found that among its member schools, less than 3% of engineering and computing courses referenced accessibility skills [1]. Accessibility is still more likely to covered in elective courses instead of core courses [5, 31, 34], thereby sending an implicit message to that accessibility is not a priority. Furthermore, inclusion of accessibility topics is strongly tied to faculty's personal commitment to the topic or their research interests [31]. Faculty want to integrate accessibility content that is specific to the area of computing they teach, which is difficult in theoretical CS courses such as algorithms and data structures [38]. Lastly, it is difficult to teach guides followed by the tech industry such as Web Content Accessibility Guidelines (WCAG) [16]. The documentation is dense and not easy to follow, making it difficult for CS students to apply in their educational and professional projects [16]. Next section discusses other challenges in following accessibility guidelines when working in the industry.

### 2.2 Accessibility in the Industry

Lazar *et al.* have found that lack of time, training, managerial support, client support pose as significant barriers to accessibility [18]. In addition, software tools are often inadequate and accessibility guidelines can be confusing to web developers [18], also confirmed by other researchers [32]. People have developed online coursework to educate professional software developers about accessibility standards, evaluation tools, and manual and automated testing [11]. But such courses can be difficult to follow alongside full-time jobs [11]. Some developers also feel that prioritizing accessibility could lead to project delays or limit creativity [3]. The effect of poor consideration toward accessibility is evident in the websites and applications! A 2023 survey by WebAIM found that 83% of web pages have low contrast text, 58% did not have alt-text for images, and 45% of the pages did not include form labels [14].

Researchers have found that software developer job postings rarely list accessibility as a required skill set [21]. Developers are not expected to possess accessibility knowledge and experience. Instead, the advocacy and education responsibilities fall largely on employees in accessibility-specialist roles [4, 21] or on developers with disabilities [27],

who are far and few in between. While large companies can still hire people with specialized skills to assist all the product teams, small companies lack the resources to do so [4]. The general lack of awareness also has a bearing on accessibility. Furthermore, people's accessibility knowledge in the industry largely comes from on-the-job training [14]. In a WebAIM survey, roughly 81% of the respondents shared that they had learned about accessibility through collaboration with colleagues [14], suggesting that we ought to look beyond college curriculums to build accessibility awareness among software developers. Patel *et al.* recommended building IDE tooling to assist developers in catching accessibility violations [28]. Others have recommended assigning and ranking severity scores to direct developers' attention to most critical accessibility issues [42].

The good news is that new web browser features that enhance page layout and design and the emergence of cross-platform UI technologies have had a positive impact on accessibility [33]. However, the effects were not planned keeping accessibility in mind. Going forward, they should be a focus of developers. For instance, one can start by providing accessible code samples [28] and inclusive UI components [9, 29] that developers can copy-paste directly. Research evidence suggests that developers often import code from the official documentation, using it as a starting point for their tasks and modifying them to meet their coding goals [22]. Furthermore, they tend to skim the documentation and are likely to miss critical accessibility information on how to make the component inclusive [22]. Thus, important concepts should be integrated through examples and information that stands out [22].

In summary, there are efforts to educate CS developers about accessibility through coursework. However, developers still struggle with applying accessibility standards. Therefore, we need to explore alternate ways to build awareness among developers entering the workforce and those already a part of the industry.

## 3 DESIGN PROCESS

Our primary goal was to build accessibility awareness among UI developers. To this end, we adopted the user-centered design process with the following steps: (1) conducting formative interviews to understand the accessibility information sought by UI developers, (2) ideating approaches for building awareness based on the interviews, (3) conducting a design workshop to refine the approaches, (4) implementing the approach selected from the design workshop, (5) evaluation study with professional UI developers to understand the effectiveness of our implementation.

We started by selecting a UI framework to scope our process. Our selection criteria was that framework should provide features to support accessibility testing and development. Additionally, it needed to be open source for us to make changes to its source code during the implementation phase. We chose Flutter [12], a UI toolkit that enables cross-platform development for Android, Windows, Linux, Mac, and the web from a single codebase. Our choice was shaped by Flutter's popularity as the leading cross-platform UI framework among developers [7]. Flutter also provides several features to support the development of accessible applications. It includes the semantics widget to customize the UI's behavior on assistive technologies. It also provides the Accessibility Guideline (AG) API which flags missing labels, small touch target sizes, and poor text contrast for accessibility testing. Lastly, Flutter and its web documentation is open source. Therefore, we could fork Flutter's GitHub repository [10] to implement our idea in a copy of its website's source code and stage the website locally. In addition, we could submit a pull request to integrate our changes into the official repository if the results were positive[2].

[2]After the evaluation study, we submitted a pull request to the Flutter team in November 2022 to integrate our changes into the official tutorial.

### 3.1 Formative Interviews

We conducted formative interviews with three different groups of people (see Table 1) to identify what kind of accessibility information developers seek during the development process and how do they acquire it. All participants were recruited through snowball sampling as we wanted to recruit people with specific skills and experience, which was hard to reach through online recruiting. The interviews were conversational and semi-structured and in nature, and lasted between 25 — 30 minutes.

Table 1. Breakdown of participants across different groups during our formative interviews

| Category | Description | Total Participants |
|---|---|---|
| Flutter Developers | Software developers who had contributed to Flutter's Accessibility Guideline (AG) API | 3 |
| Flutter Users | Software developers who currently used Flutter's accessibility features, including the AG API, for testing and debugging their application | 3 |
| Accessibility Experts | Professionals within tech companies who supported software engineering teams in tool selection, accessibility compliance, and were engaged in accessibility advocacy | 4 |

We first interviewed three software engineers who had developed and contributed to Flutter's Accessibility Guideline (AG) API. Our goal from these interviews was to understand what led the developers to design the AG API, how they selected the accessibility principles to guide the API design, and how do Flutter users utilize the API in their workflows. The first author took detailed notes about the creation of the API, how it facilitated unit testing, and how it was documented for use by all Flutter users.

Next, we interviewed three professional developers who were advanced Flutter users and used Flutter's accessibility features, including the AG API, to test and debug their applications. These interviews complemented the findings from the previous interviews and helped us understand how Flutter users discovered and used its accessibility features, including customization of the AG API. We also asked the developers to share code snippets demonstrating their use of the AG API and UI screenshots to understand how they captured and debugged accessibility issues. We took detailed notes for future analysis. Since we elicited highly specific examples on the use of Flutter's AG API and accessibility features, we also recorded these interviews. Participants provided written consent to the recording via a form prior to interview.

Lastly, the research team conducted interviews with four accessibility experts who had extensive experience working with developers and software engineering teams in technology companies. These interviews helped us look beyond the needs of Flutter users and developers and identify the information needs of the programming community more generally. We recruited people who supported development teams in complying with accessibility for their applications, helped teams select accessible programming tools and frameworks, and advocated for following best practices regarding accessibility in their organizations. During these interviews, we focused on understanding the accessibility content they used to educate engineering teams. We also elicited their perspectives on how to build accessibility awareness in the programming community.

### 3.2 Initial Findings

We wrote analytical memos [37] after each set of interviews and open coded the transcripts to analyze the data collected from the interviews. We found that the accessibility information that developers and testers often seek in official documentation can be organized into four categories:

(1) **Assistive technologies' (ATs) set up and explanation**: A primer on different types of ATs, such as screen readers, switch access, etc., including how to activate and set up the ATs.
(2) **UI behavior on ATs**: Preview of the expected behavior of UIs on different ATs. Our interview participants revealed that images can help flag accessibility issues such as poor contrast and small font size through UI screenshots.
(3) **Accessibility principles**: This includes the common accessibility guidelines developers should keep in mind during development and testing. Typically, documentation of UI frameworks do not list all the recommendations under WCAG 2.1. For users with visual impairments, we noted the official documentation of React Native, Angular, Flutter, and Android emphasized checking for contrast, touch target size, target labels, and alt-text
(4) **Accessibility testing**: Sample code and explanations to demonstrate API use and accessibility testing through automated frameworks such as Selenium, Espresso, etc.

### 3.3 Design Workshop

Drawing on the findings from the interviews and prior research, the research team developed the following six ideas:

(1) The skeleton app that gets created for each new Flutter project includes default unit tests. These unit tests can be modified to demonstrate the use of the AG API and promote accessibility testing.
(2) Prompting developers to write code that meets accessibility requirements through IDE tooling.
(3) Sprinkle the official documentation with accessibility information
(4) Highlight accessibility in code samples on DartPad, a web-based code editor that offers Flutter code samples for developers to edit and explore without installing the prerequisites.
(5) Preview assistive technology output through IDE tooling.
(6) Show expected behavior of UI on ATs through screen captures, video recordings, etc.

We conducted a 90 minutes design workshop with ten participants to evaluate each idea. Three of the workshop participants, including the moderator, identified as women; the rest of the participants identified as men. One participant identified as a person with visual impairment; all other participants identified as sighted. We presented examples and designs to explain the ideas. For instance, we included screenshots from Inclusive Component's website [29] to describe idea #4. All participants possessed experience working in technology companies and had contributed to programming frameworks and languages. The participants included software developers, technical writers, accessibility experts, and researchers with background in Human-Computer Interaction (HCI) and accessibility. One AG API developer and an accessibility expert who participated in the formative interviews were part of the workshop. We used snowball sampling to invite participants to the workshop.

The initial half an hour of the workshop was spent on a brief ice breaker followed by an explanation of the research project, presentation of findings from the formative interviews (see §3.2), and the goals of the research team. For the remaining one hour, the participants spent approximately ten minutes to discuss each idea. The discussion was moderated by the first author to identify the technical feasibility of the idea's implementation as well as its potential in building awareness about accessibility among developers. One member of the research team took detailed notes to

facilitate analysis and implementation later on. Drawing on the workshop discussion, the research team combined ideas #3 and #6 — sprinkling the documentation with accessibility with the opportunity to preview the UI's performance on screen readers. We call our intervention the Blended Approach (BA) toward accessibility documentation.

## 4 BLENDING ACCESSIBILITY IN OFFICIAL DOCUMENTATION

We chose Flutter's onboarding tutorial (part one) to test the effectiveness of our design idea. The tutorial was divided into two parts. Part one was a single webpage and existed as part of the official documentation and therefore open source and editable; part two linked participants to a Google codelab and was not editable[3]. Our rationale for choosing the tutorial was that it was likely to get relatively more traffic compared to other pages in the documentation and we could introduce accessibility early in the development process. In addition, developers were likely to implement the steps outlined in tutorial to gain hands on experience. Thus, tips and suggestions on creating an accessible application were more likely to get incorporated into developers' workflows.

We started by creating accessibility content for each category of information identified through our formative interviews (see 3.2). The research team members met multiple times and consulted the workshop participants to discuss the length and representation (e.g., text, video, image, etc) of each piece of content. Our goal was to not detract from the primary purpose of the tutorial. We wanted the accessibility content to be subtle, in essence, *blended* into the existing text and not seem out of place or forced. Below we briefly describe each of the 7 additions to the tutorial, including the category they map to (see §3.2):

(1) The tutorial opened with learning objectives, where the first bullet point listed the platforms on which the application would work. We followed this point with a second bullet point suggesting that the tutorial could also be tried with screen readers and cross linked to the videos on turning on screen readers (Category 1; ATs set up and use)
(2) The learning objectives section concluded by stating that part 2 of the codelab would focus on adding interactivity and navigation to the application. We modified the statement to say that part 2 would also focus on meeting accessibility requirements. (Category 3; accessibility principles)
(3) The tutorial explained several features of Flutter and Dart as a bulleted list. We added a final bullet to the list that linked to the documentation on Flutter's semantics widget (Category 3; accessibility principles)
(4) The tutorial contained an explanation of Pascal case, highlighting it in a blue box. We added another box to say how Pascal case enabled clear pronunciation of compound words on screen readers (Category 3; accessibility principles)
(5) Part one concluded with a screenshot of what the iPhone version of the final application would resemble. We recorded a screen capture of the application on Android TalkBack and placed the video next to the screenshot. This way, people could experience the application's performance on a screen reader (Category 2; UI behavior on ATs)
(6) We embedded short YouTube videos on how to turn on TalkBack on Android and VoiceOver on iPhone respectively. The tutorials were placed after completion of the first component of the tutorial to prompt readers to try out their code on screen readers if they wished (Category 1; ATs set up and use)

[3]Codelabs are guided tutorials created by Google Developers hosted on https://codelabs.developers.google.com/. While the codelab samples are available on GitHub, the website is not open source and cannot be edited to include new content

(7) At the end of the tutorial we included links for exploring the Flutter SDK further. We added a link to testing accessibility in Flutter mobile apps and updated the accessibility page to include examples on how to use the AG API. The examples were based on the tutorial code (Category 4; accessibility testing)

Figure 1 shows screenshots of the content we incorporated into the tutorial. We forked the flutter/website repository on GitHub [10] and followed the steps outlined on the repository page to integrate our changes into the tutorial. The instructions list how to use Firebase to stage the edits within one's copy of Flutter documentation. Perusing these instructions, we hosted the modified Flutter website, which included the tutorial and the accessibility page, on a different URL and shared the website's link with our participants during the evaluation study.

## 5 EVALUATION STUDY

We conducted an evaluation study with 11 front-end/full-stack developers to understand the response and perceptions regarding the accessibility content in the tutorial.

### 5.1 Pilot Study

We conducted a pilot study with two sighted developers to examine if our content was understandable. In addition, we also requested an accessibility expert to review the accessibility content we created. We made minor changes to the content based on the feedback we received. For instance, we increased the volume of the preview video to be more audible based on suggestions. The pilot study also helped us modify certain aspects of the study design, such as think aloud protocol's instructions and the follow-up interview questions.

### 5.2 Participants

We created a screening questionnaire to recruit participants that met our eligibility criteria. As part of the survey, we included questions about participants' prior experience in programming, Flutter, and WCAG. To be eligible, participants had to be 18 years of age or older and work as front-end or full stack developers.

We specifically recruited developers who reported having little to no awareness of Flutter. The criteria ensured that participants would not contrast our changes with their prior knowledge of Flutter's onboarding experience. We also filtered out participants who listed having intermediate or advanced WCAG experience. Any accessibility content was highly likely to get noticed by developers with extensive knowledge of accessibility and they might react to them positively. Furthermore, our goal was to build accessibility awareness among developers who may lack the domain knowledge. To avoid giving away the purpose of the study, we included additional questions about other topics such as unit testing. This was done to keep participants from thinking the study would focus on accessibility.

We compensated each participant with 100 USD. 9 participants identified as men, one participant identified as women, and one preferred not to share their gender in the screening questionnaire. All participants fell between the age range of 18 – 60 and had more than three years of programming experience. Table 2 lists each participants' programming experience, job role, and level of familiarity with WCAG.

### 5.3 Study Design

Each study session was conducted remotely over Google Meet and lasted approximately 90 minutes. We informed our participants that the study's purpose was to get feedback on the contents of the tutorial page. We shared the link to the staged site via chat and asked participants to open the tutorial on their end and screenshare their browser.

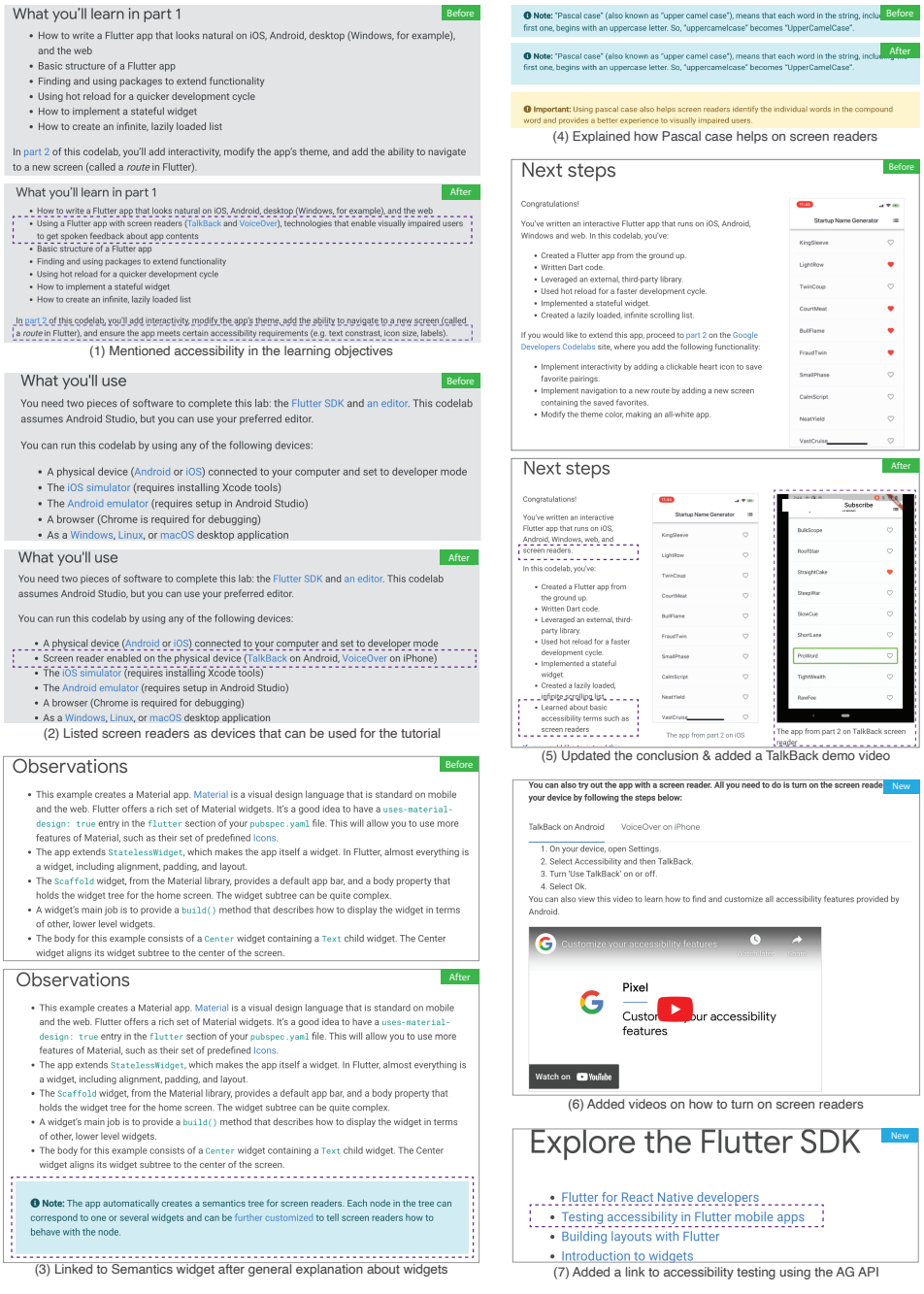


Fig. 1. Accessibility content added to the getting-started tutorial corresponding to the information categories identified from the formative interviews

Table 2. Demographic characteristics of the participants. Age and programming experience reported in years

| ID | Age | Gender | Job Role | Prog. Experience | WCAG Familiarity |
|---|---|---|---|---|---|
| P1 | 31–40 | M | Software Developer | 5–10 | Somewhat familiar |
| P2 | 24–30 | Prefer not to say | Software Developer | 3–5 | Somewhat familiar |
| P3 | 41–50 | M | Software Developer | 10+ | Not very familiar |
| P4 | 18–23 | M | Software Developer | 3–5 | Somewhat familiar |
| P5 | 18–23 | M | Software Developer | 3–5 | Somewhat familiar |
| P6 | 51–60 | M | Tech Lead | 10+ | Somewhat familiar |
| P7 | 24–30 | M | Tech Lead | 5–10 | Somewhat familiar |
| P8 | 24–30 | W | Software Developer | 3–5 | Not very familiar |
| P9 | 41–50 | M | Software Developer | 10+ | Somewhat familiar |
| P10 | 41–50 | M | Tech Lead | 10+ | Somewhat familiar |
| P11 | 24–30 | M | Software Developer | 5–10 | Somewhat familiar |

We told the participants that they could explore the tutorial in any manner, including clicking on links, videos, and resources. They were asked to think aloud as they read the content. We emphasized to them to share their thoughts on anything they found *interesting* or *irrelevant*. The think aloud approach captured whether participants truly noticed the accessibility content in the tutorial as well as their thoughts on the content. We also told the participants that they were not required to install Flutter or write code to create the tutorial application. Our rationale was that installation, writing, and debugging the tutorial code would make the study sessions significantly longer and leave us with limited time to gather participants' feedback on the accessibility content.

We collected written consent from all participants prior to the start of the study. We recorded the screenshare and the conversation, which was auto-transcribed by a third-party transcription application. As participants browsed the tutorial, the study moderator noted down participants' verbal comments as well as actions such as selections, hovering, clicks, etc. These actions also communicated the parts of the tutorial that caught their attention.

After participants finished going through the tutorial, we presented them with a Google form that comprised three questions. They were asked to (1) list three things that stood out to them in the tutorial, (2) list three things they felt was irrelevant, and (3) select their impressions of Flutter from a list of nine adjectives. The final multiple choice question was inspired by Microsoft's Desirability Toolkit [25]. The purpose of the form was to gather, without priming, if participants had noticed the accessibility content and whether it had a bearing on their perceptions about the tutorial and Flutter.

Next, we conducted a semi-structured interview to collect qualitative feedback about the tutorial's length and content. In the initial questions, we avoided priming the participants to see if they brought up accessibility content without being prompted. After participants had described their thoughts on the tutorial, we disclosed the purpose of the study and followed up with specific questions about accessibility. We asked participants to give detailed feedback about the placement (e.g., keep it in the tutorial or remove it), representation (e.g., text, video, or audio), and length for each content.

### 5.4 Analysis

We annotated the video recordings to note participants' explicit reactions to each piece of content we had added. We only counted instances where participants exclaimed or commented on the content. We ignored cases such as participants hovering over the content but not reacting explicitly to avoid false positives.

We relied on inductive coding to analyze the think-aloud data and the interviews. We developed six initial themes and reorganized them into three high-level themes. The high-level themes inform the findings section of our paper.

Participants' responses to the Google form were unprompted. We counted instances of accessibility mentions for each question and analyzed their comments along with the interview data. We discuss the form responses in the findings section reporting on participants' unprompted reactions to accessibility content.

## 6 FINDINGS

### 6.1 How did participants react to the accessibility content?

In this section, we describe participants' unprompted reactions to accessibility content in the tutorial. Table 3 summarizes the pieces of content that were noticed by participants.

Most participants responded positively, using descriptors such as ''*awesome*'' (P2), ''*great*'' (P6), and ''*pretty nice*'' (P7), when they noticed the snippets of accessibility content we had added to the tutorial. As described in 4, the learning objectives section at the beginning of the tutorial mentioned how to run the tutorial application on screen readers and gave a brief explanation about screen readers. It was noticed by 8 participants and most of them reacted favorably upon reading the line:

> *I like how immediately you're highlighting accessibility features right here [under 'what you'll learn in part 1']. That's becoming, not that that was never not important, but it's becoming more important as I feel like more developers are trying to be more, you know, accessibility focused, okay?* —P4

To ensure natural reading behaviors, we had informed participants that they were welcome to click on any links or videos they wanted. We observed that several participants were curious and explored the content we had included. For instance, when P2 read the brief explanation we had added about Flutter's semantics widget, he clicked on the link to read more about the widget and how it could be used to modify the accessibility of Flutter applications. He had clarifying questions and remarks about the widget's functionality. Participants also tended to skim the tutorial. They did not read every detail or go through the tutorial sequentially, modeling the realistic behavior of developers when they browse documentation [22]. A few participants missed the early mentions of accessibility under learning objectives, which contextualized the rest of the information we had added to the tutorial. When they directly noted content in the middle of the tutorial such as the information block on how Pascal case enabled accurate pronunciation on screen readers (see 1) or the instructional videos on how to turn on screen readers on Android and iOS, participants expressed confusion:

> *Interesting [on seeing instructional videos on how to turn on screen readers]! I'm not sure why this would be here. It sort of feels out of place. And so I'm a little confused as to why this would be here* —P5

After remarking on the instructional videos, P5 continued skimming the tutorial. The tutorial concluded with a summary of the learning goals, which he read point by point, and noticed that the tutorial focused on ''*ensuring the app meets basic accessibility requirements*'' (P5). Upon realizing that accessibility was a focus of the page, he mentioned the ''*screen reader video makes sense*'' and was appreciative of the tutorial:

> *'I think that's really driving on the point to make this app accessible, even from scratch. That's cool! I like that, you know, accessibility requirements, because normally when you code, you it's not one of those things that you really think about and so it's good to see that it's included here if you know when you're learning how to build it.* —P5

One of our research goals through BA was to communicate to developers that they can bring forward accessibility in the development process. P5's unprompted quote above suggests the approach can be successful in priming developers to make accessibility a priority.

We were also concerned that participants may find the accessibility content extraneous, ultimately leading them to feel that the tutorial was too lengthy. As mentioned in section 5.3, we presented participants with a short form to list things that 'stood out to them' or felt 'irrelevant to the tutorial.' Only two participants (P5 and P8) reported that they felt the information about screen readers was not pertinent to the tutorial. P5, however, similar to his think aloud comments discussed earlier, wrote that the mention of accessibility under learning objectives changed his mind. P8 stated in the form that she found the tutorial to contain ''*a lot of text*'' and commented during the task that the text was not easily ''*digestible.*'' We believe this may have shaped her perception of the accessibility content, specifically the instructional videos on screen readers, which occupied relatively higher real estate and was likely the reason it was noticed by 10 out of 11 participants. P8 may have felt the videos added to the tutorial length. P8 also mentioned that information blocks caught her attention. She made sure to read them and exclaimed ''*I like this*'' upon seeing the snippet on Pascal Case. Her behavior suggests that some developers may go through accessibility information when it is presented as a small, independent blocks of content. Besides P5 and P8, P7 reported the screen reader videos as irrelevant. However, he wanted the videos to be supplemented with additional information on assistive technologies, also confirmed by his think aloud comments:

> *While useful, the mention of TalkBack/Voiceover was short and did not have much follow up. The tutorial would have been just as useful (from a first time flutter perspective) without it.* —P7

None of the participants commented on any other accessibility content being unessential to the tutorial. Additionally, three participants' form responses explicitly stated that they liked the accessibility content.

### 6.2 What were participants' thoughts about BA?

The previous section discussed participants' perceptions of accessibility content without the research team prompting or priming them. In this section, we highlight findings from the interview after we disclosed the purpose of the study and asked participants to share their thoughts on BA.

Consistent with participants' unprompted comments, many participants said they liked BA for learning more about accessibility concepts, which was difficult to discover in the documentation on one's own:

> *I'll be honest, I haven't had too much experience [with accessibility]. So I think I've worked a little bit with it in in web, you know, with ARIA things. But I want to know more but it feels like it's a bit harder to find sometimes. So I think that's why it's very important that it's being promoted here within the basic tutorial that people might follow.* —P4

Participants' responses validated our insights from the formative interviews and the design workshop. We had hypothesized that providing a preview of the application's functionality on assistive technologies would give developers a glimpse of how users with disabilities experience the application. Participants shared that the preview video included in the tutorial was informative of screen reader behaviors:

Table 3. Summary of each piece of content that participants explicitly noticed. Rightmost column mentions the total number of accessibility snippets noticed by each participant. Last row mentions the total number of participants who noticed the accessibility snippet corresponding to the column

| ID | Learning Objectives | Semantics Widget Link | Pascal Case | Screen Reader Instructions | TalkBack Preview | AG API Link | Total |
|---|---|---|---|---|---|---|---|
| P1 | Yes | Yes | Yes | Yes | No | No | 4 |
| P2 | Yes | Yes | Yes | Yes | No | Yes | 5 |
| P3 | No | Yes | No | No | No | No | 1 |
| P4 | Yes | No | Yes | Yes | Yes | No | 4 |
| P5 | No | No | No | Yes | Yes | No | 2 |
| P6 | Yes | No | Yes | Yes | Yes | No | 4 |
| P7 | Yes | No | Yes | Yes | No | No | 3 |
| P8 | Yes | No | Yes | Yes | Yes | No | 4 |
| P9 | No | No | Yes | Yes | Yes | Yes | 4 |
| P10 | Yes | No | No | Yes | No | No | 2 |
| P11 | Yes | Yes | Yes | Yes | No | No | 4 |
| Total | 8 | 4 | 8 | 10 | 5 | 2 | -- |

> *I have not really played with screen readers even though I know what they do. I was curious to see how it [the tutorial application] would behave. The video was really to the point. Just a few seconds long, meaningfully obvious! I liked it!* —P6

Similarly, instructions on how to turn on screen readers and try out the application enabled participants to discover information that they otherwise were unsure of finding on their own. Some participants commented that they would want instructions for using the screen reader and accessibility features on platforms that they used as part of their development workflows such as Windows operating system and Google Chrome browser.

Participants described instances from their professional lives that helped them to learn about certain accessibility principles. For instance, a few participants shared that they had learned about the importance of color contrasts and staying ''*away from certain color palettes*'' (P3) to ensure the UI design was colorblind safe. In some cases, these resources were not archived for use after the project completion, preventing participants from referencing them again. Furthermore, the project instructions lacked explanations on why these principles were important, which prevented them from internalizing their takeaways for future projects. Having accessibility principles blended into the documentation at various points gave participants the confidence that they could access the information whenever they liked.

Participants also shared personal accounts of how they had become interested in accessibility and were trying to be more mindful during development to build more accessible and inclusive applications:

> *I first became interested when I went to visit my grandmother and saw that she was having trouble reading her screenshot, to make the font super large. And I was like, well that makes sense. Not everyone can use the computer in the same way. So then I felt like I should probably pay attention to that a little bit more* —P2

We noted that participants' personal experiences influenced the kind of accessibility content they wanted blended into the documentation. For instance, P1 shared that he often has to ''*pinch and zoom*'' to read text on his phone or reduce the glare at night for better readability:

> *There should be more ways other than having a screen reader. Like if I am scrolling my phone and if I'm a disabled, if I need accessibility options, screen reader is one way but there are other ways. So it would be nice to include those [...] So it didn't tell me how a user could increase the font size. [...] If I do invert colors, how does Flutter react to it?* —P1

Only P5 reported learning about accessibility formally as part of a web development course he had taken during his undergraduate degree. The course introduced him to tools that enabled him to ''*get an accessibility score and it kind of looked at button colors*'' (P5). But he shared that the topic was not covered in enough detail in the course. Furthermore, his current job does not require him to incorporate accessibility in every development project. According to him, BA centred the importance of accessibility for the programming community:

> *I probably have seen hundreds of tutorials and, you know, accessibility is never a thing! As coders we tend to put that to the side, it becomes an afterthought when we're coding [...] Even in school, it's not really a huge focus. So definitely becomes an afterthought in the real world.* —P5

A similar thought was echoed by P8. She was the only participant who preferred a single page dedicated to discussing accessibility over BA. However, she shared the Flutter tutorial suggested to her that accessibility could be brought forward in one's workflow, including in small projects:

> *I always assumed it's [accessibility] kind of separate. Like it's something you add on later. But then now, when I'm reading flutter, it's like, 'oh, it's integrated into it', even when you're building your first Flutter app. It's like a key part of it!* —P8

We also asked participants if they had encountered a similar approach to accessibility in other documentation. Only one participant, P11, mentioned that he had seen Tailwind [39] prioritize accessibility in its documentation. However, accessibility was not ''*as at the forefront*'' (P11) as achieved through BA. We also noted that several participants perceived Flutter as being more inclusive and a more accessible framework relative to other frameworks:

> *If I was shopping around for UI frameworks, its nice to see right away this supports my accessibility use cases versus the other might not. That might be enough to kind of sway me one way or the other* —P7

In conclusion, almost all participants appreciated the BA and felt it could be used for ''*educating people*'' and ''*building the acceptance*'' (P6) for integrating accessibility earlier in the development workflow. Only P8 remarked that she would prefer all accessibility-related content to have a dedicated page, similar to the industry norm.

### 6.3 Participants' Feedback on Accessibility Content

This section reports on participants' general feedback as well as specific recommendations on each accessibility content.

As mentioned in section 4, we updated the learning objectives and the conclusion to state that the tutorial would focus on certain accessibility principles. In relation to these, participants recommended that we should also update the tutorial title to emphasize accessibility:

> *'Flutter part 1', you know somewhere there, you could have potentially used the word 'accessible' or something to be very clear to me that we are not only creating a standard app but it's also accessible. So mentioning it really high up, putting some importance on it. Putting it in bold would be huge! Because I know a lot of times as coders we might like to skim through a page quickly enough to get the most important content.* —P5

Furthermore, participants suggested explaining the meaning of *accessibility* because some developers may be completely unaware of the term. We concluded a similar recommendation based on the study sessions with two participants (P3 and P10). Through follow-up questions in the interview, we noted that P3, despite being familiar with terms like ARIA and assistive technologies, used the word *accessible* to imply that the tutorial's language was easy to follow for non-native English speakers. P10, on the other hand, had no prior knowledge of accessibility terms. For such participants, a definition at the beginning, would help establish consistent vocabulary. However, participants also advised against including accessibility content in long tutorial videos that covered multiple topics. They felt that videos were useful only when they were short and completely focused on accessibility:

> *I opened or two [videos of the tutorial].I think one of them was 45 minutes. I mean what we can do is have these one or two minutes, short videos, max 3 minutes. And just showing the capability of how you do it and then giving the user a link that would take them to different documentation on how to do it, along with the 45-minute video if possible.* —P1

Participants feedback was also shaped by their existing knowledge and experiences. For example, participants with more web development experience wanted to go through the ''Write your first Flutter App on the Web'' tutorial. They suggested that the web-based tutorial should include snippets on ARIA labels and how to use the browser developer tools, a tool suite included within all major web browsers, for accessibility testing. It is also worth noting that a few participants explicitly recommended against creating a tutorial solely focused on accessibility. They felt that ''*people might skip it*'' (P6) when it is suggested as a series of steps after the main tutorial and may be too much to do in one go. Instead, they felt including tailored content across multiple tutorials was a better approach.

Participants liked the use of colored information blocks to call out attention to accessibility principles, also confirmed by the think-aloud data. We had defaulted to the tutorial's green color when creating our information blocks. Two participants suggested using yellow to distinguish accessibility tips from other tips and to communicate that not following the tip will not cause breakdowns to the app:

> *I think yellow is a good color for it [accessibility principles] because it's important. It's not going to cause you problems if you don't do it [accessibility instructions] properly, but it is important to do it this way.* —P4

Similarly, P5 mentioned using a ''*badge, or tag or an icon*'' to delineate the information blocks on accessibility from other blocks. Participants also stressed on keeping the content concise and linking to detailed explanations to facilitate skimming and additional reading for the more curiously-inclined, like we had done for the AG API and the semantics widget. Lastly, participants suggested using accessibility-focused examples and code samples in pages that explained important programming concepts such as unit testing and debugging to promote the AG API:

> *You also add it to, because not everyone, might click the accessibility link [to AG API at the bottom of the tutorial] but I think everybody would click testing and debugging while going through something. So like maybe including this inside there, since it is about testing as well, but we're making more people aware.* —P8

They also recommended using images and GIFs to show failed unit tests. For example, a screenshot of an app with poor contrast could be used to demonstrate lack of compliance to contrast guidelines.

## 7 DISCUSSION

In this section, we first summarize the effect of BA on developers and then present a framework for documenting accessibility in UI frameworks and libraries.

### 7.1 Effect of BA on Developers

Our findings show that the addition of short snippets of accessibility content was not perceived as making the tutorial lengthy and did not disrupt developers' reading flow. The accessibility mentions were received positively. In fact, developers wanted to learn more and shared examples of additional accessibility-related content they would like to see in the documentation. Our participants shared that they did not always know how to find accessibility information that is relevant to the programming technologies they have chosen. With BA, developers felt that accessibility was easier to look up in the documentation and could be integrated into the programming workflow from the scratch. It could even be a consideration in introductory resources such as tutorials. Furthermore, our study showed that most developers prior experience with accessibility comes from personal and professional interactions. Only P5 shared that he had received instruction on accessibility in a web development college course. Combined with the survey results by WebAIM [13], which showed that developers generally gain accessibility experience from colleagues, our research demonstrates that BA can be a viable solution for building accessibility awareness in the industry.

BA shaped participants' perceptions positively about Flutter. They felt that by emphasizing accessibility and highlighting its features such as the semantics widget and the AG API, Flutter demonstrably cared for inclusion. Participants shared that seeing the accessibility features up front would sway their decision to use the framework. The findings reveal two important things. First, developers are unlikely to discover features such as APIs, accessibility tools, etc if it is all compiled in the single place. Thus, UI frameworks may see more adoption if they make their accessibility features discoverable through BA, especially as the industry grows more inclined towards offering accessible products to end users [2]. Second, BA needs to be exercised with caution. Pandey *et al.* have warned that developers tend to overestimate the accessibility capabilities of UI frameworks. BA should be used as means for building awareness and not for advertisement. It should serve to educate developers on how to adopt the right series of steps as they write and debug code and to bring accessibility forward in their development workflows, which ultimately leads to fewer accessibility issues [26] and can be beneficial in teams that cannot afford to hire accessibility specialists [4, 21]. It should *not* suggest that products team can forego accessibility testing.

### 7.2 Framework for Adding Accessibility to Documentation

We outline the framework that documentation authors and teams working on UI frameworks can use for building accessibility awareness among their users.

(1) **Discoverable**: BA strives to make content across each of the four categories of accessibility discoverable to developers. The goal is to enable multiple routes to accessibility information instead of a single dedicated place for quick reference. We suggest creating content such that it places accessibility at par with other important programming concepts such as security, performance, and UX. The suggestion is akin to the recommendation made by CS education researchers who advise against introducing accessibility through electives and instead propose making it a part of the core CS topics [5, 31, 34].

(2) **Repeatable**: It is essential to repeat information across pages because developers tend to skim documentation [22] and are likely to miss the information if it is not repeated. We point readers to the examples shared by our

participants. They suggested highlighting the use of AG API on Flutter's testing and debugging web pages besides presenting the details on the dedicated accessibility page.

(3) **Understandable**: Each piece of content integrated into the documentation should be easy to read, understand, and internalize as a lesson. For instance, we included a two-line long information block on how Pascal case supports accurate pronunciation of compound words on screen readers. Upon reading it, several participants commented that they were previously unaware of the accessibility use case of Pascal case but would remember it now. We encourage documentation writers and framework developers to utilize similar examples to call attention to accessibility. We also suggest utilizing different mediums such as images, GIFs, videos, information blocks, code comments, and code samples to present the content.

(4) **Non-disruptive**: BA should not detract from the main topic. The strength of the approach lies in emphasizing that accessibility can be built into development workflows from the start without extra effort. To this end, the blended content on accessibility should not appear as additional steps to execute after the fact. Furthermore, certain programming processes can only be performed in a sequence and cannot be performed by software developers. For example, quality analysts and accessibility specialists test for consistent behaviors across all operating systems and assistive technologies as the final part of the software release process. While these topics are essential and often covered in UI documentation, they should be presented as standalone topics rather than blended into existing pages to avoid the risk of developers placing too much confidence into the accessibility of the UI technologies they have chosen.

(5) **Tailored**: Prior research has shown that accessibility standards are confusing and difficult to follow [16, 18, 32]. Drawing on our findings and related work, we recommended tailoring the content to each page. For instance, a web page discussing mobile app development should present topics that improve accessibility for mobile applications. On the other hand, a web page dedicated to web development should discuss ARIA, browser developer tools, etc. Tailored content would not only prevent developers from getting overwhelmed but also allow them to tie the accessibility concepts with their existing programming knowledge.

The above framework details *how to use* the BA. The four accessibility categories we had identified through the formative interviews (see section 3.2) detail the kinds of accessibility content to create when using BA: (1) ATs set up and use, (2) UI behavior on ATs, (3) accessibility Principles, and (4) accessibility testing. Finally, as shown in our implementation, we utilized different mediums such as images, videos, and screen captures to make the content easy to consume without distracting from the tutorial. We recommend exploring different mediums when employing BA.

### 7.3 Limitations

We did not require participants to create the tutorial application during the study. Participants might have responded to the accessibility content differently if they had written the code and tried out the application. They might have gone through the tutorial content linearly and paid more attention to the text, thereby noticing the snippets they missed while skimming. Thus, the results may look different if developers were asked to code while they read the tutorial. Future work should compare developers' awareness as they reference the documentation while coding, which will also prompt considerations for incorporating BA into IDE tooling. However, it is essential to note the challenges of conducting a summative evaluation of BA. Despite developers receiving exposure to accessibility through BA, its effect on their awareness might be delayed, which could complicate measurements.

All of our participants were US-based. The Americans with Disabilities Act [41] requires federal applications and websites to be accessible in the US. Therefore, our participants may be more aware about accessibility requirements compared to developers in other countries where engineering teams are not legally required to enforce accessibility in their applications. Studies with developers from other countries may yield insights into how to adapt BA when legal and cultural landscapes around accessibility differ.

Lastly, we scoped our content to inform developers on how to make the application accessible for users with visual impairments. Future work should examine how to create and incorporate accessibility content for all disabilities.

## 8 CONCLUSION

UI frameworks and libraries typically reserve a single place for accessibility-related information in their documentation. This approach makes it difficult for UI developers to discover accessibility information and apply it in their development workflows. We present the blended approach (BA), a novel way of documenting accessibility. BA recommends integrating short snippets on accessibility through high-traffic pages of any UI framework's documentation. Our implementation in Flutter's getting-started tutorial and evaluation with 11 UI developers suggests that the approach can help developers explore the topics of accessibility principles, assistive technologies, accessibility testing, and UI behavior on ATs without compromising the underlying page's perceived readability and length. Through our research, we derive a framework that others can use to improve accessibility documentation in their UI technologies and build awareness among their target programming audience. Future work should include a summative evaluation to examine BA's effect on developers' awareness when accessibility is blended throughout the documentation.

## ACKNOWLEDGMENTS

We thank Victor Tsaran and JaYoung Lee for their input and advice throughout the research process. Khanh Nguyen, Brett Morgan, and Shams Zakhour's detailed instructions enabled us to implement and test our ideas. We are grateful to Steve Oney and Vaishnav Kameswaran for their feedback on the writing. Parag Vaswani, Prajakta Nakate, Rajashree Padmanabhi, and Arjun Vithalkar helped stitch the sections together with their humor and patience. Lastly, we thank all our participants, without whom this research would not have been possible.

## REFERENCES

[1] Teach Access. 2019. Why teach accessibility? Fact sheet. https://teachaccess.org/resources/fact-sheet-why-teach-accessibility/

[2] Teach Access. 2023. Teach Access. Bridging the gap between accessibility and education. https://teachaccess.org/

[3] Humberto Lidio Antonelli, Sandra Souza Rodrigues, Willian Massami Watanabe, and Renata Pontin de Mattos Fortes. 2018. A Survey on Accessibility Awareness of Brazilian Web Developers. In *Proceedings of the 8th International Conference on Software Development and Technologies for Enhancing Accessibility and Fighting Info-Exclusion* (Thessaloniki, Greece) *(DSAI '18)*. Association for Computing Machinery, New York, NY, USA, 71–79. https://doi.org/10.1145/3218585.3218598

[4] Shiri Azenkot, Margot J Hanley, and Catherine M Baker. 2021. How Accessibility Practitioners Promote the Creation of Accessible Products in Large Companies. *Proceedings of the ACM on Human-Computer Interaction* 5, CSCW1 (2021), 1–27.

[5] Catherine M. Baker, Yasmine N. El-Glaly, and Kristen Shinohara. 2020. A Systematic Analysis of Accessibility in Computing Education Research. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education* (Portland, OR, USA) *(SIGCSE '20)*. Association for Computing Machinery, New York, NY, USA, 107–113. https://doi.org/10.1145/3328778.3366843

[6] Kimberly Edginton Bigelow. 2012. Designing for Success: Developing Engineers Who Consider Universal Design Principles. *Journal of Postsecondary Education and Disability* 25, 3 (2012), 211–225.

[7] Jet Brains. 2021. The State of Developer Ecosystem 2021. https://www.jetbrains.com/lp/devecosystem-2021/miscellaneous/

[8] Robert F Cohen, Alexander V Fairley, David Gerry, and Gustavo R Lima. 2005. Accessibility in introductory computer science. *ACM SIGCSE Bulletin* 37, 1 (2005), 17–21.

[9] Michael Crabb, Michael Heron, Rhianne Jones, Mike Armstrong, Hayley Reid, and Amy Wilson. 2019. Developing Accessible Services: Understanding Current Knowledge and Areas for Future Support. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems* (Glasgow, Scotland Uk) *(CHI '19)*. Association for Computing Machinery, New York, NY, USA, 1–12. https://doi.org/10.1145/3290605.3300446

[10] Flutter. 2023. GitHub - flutter/flutter: Flutter makes it easy and fast to build beautiful apps for mobile and beyond. https://github.com/flutter/flutter

[11] Greg Gay, Naza Djafarova, and Leonora Zefi. 2017. Teaching Accessibility to the Masses. In *Proceedings of the 14th International Web for All Conference* (Perth, Western Australia, Australia) *(W4A '17)*. Association for Computing Machinery, New York, NY, USA, Article 15, 8 pages. https://doi.org/10.1145/3058555.3058563

[12] Google. 2022. *Flutter*. Google, Mountain View, CA. https://flutter.dev/

[13] Policy Institute for Disability Research and Practice. 2023. Survey of Web Accessibility Practitioners #3 Results. https://webaim.org/projects/practitionersurvey3/#learning

[14] Policy Institute for Disability Research and Practice. 2023. The WebAIM millionthe 2023 report on the accessibility of the top 1,000,000 home pages. https://webaim.org/projects/million/

[15] Saba Kawas, Laura Vonessen, and Amy J. Ko. 2019. Teaching Accessibility: A Design Exploration of Faculty Professional Development at Scale. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education* (Minneapolis, MN, USA) *(SIGCSE '19)*. Association for Computing Machinery, New York, NY, USA, 983–989. https://doi.org/10.1145/3287324.3287399

[16] Claire Kearney-Volpe, Devorah Kletenik, Kate Sonka, Deborah Sturm, and Amy Hurst. 2019. Evaluating Instructor Strategy and Student Learning Through Digital Accessibility Course Enhancements. In *Proceedings of the 21st International ACM SIGACCESS Conference on Computers and Accessibility* (Pittsburgh, PA, USA) *(ASSETS '19)*. Association for Computing Machinery, New York, NY, USA, 377–388. https://doi.org/10.1145/3308561.3353795

[17] Amy J Ko and Richard E Ladner. 2016. AccessComputing promotes teaching accessibility. *ACM Inroads* 7, 4 (2016), 65–68.

[18] Jonathan Lazar, Alfreda Dudley-Sponaugle, and Kisha-Dawn Greenidge. 2004. Improving web accessibility: a study of webmaster perceptions. *Computers in human behavior* 20, 2 (2004), 269–288.

[19] Stephanie Ludi. 2002. Access for everyone: introducing accessibility issues to students in Internet programming courses. In *32nd Annual Frontiers in Education*, Vol. 3. IEEE, New Jersey, NJ, USA, S1C--S1C. https://doi.org/10.1109/FIE.2002.1158617

[20] Stephanie Ludi. 2007. Introducing Accessibility Requirements through External Stakeholder Utilization in an Undergraduate Requirements Engineering Course. In *Proceedings of the 29th International Conference on Software Engineering (ICSE '07)*. IEEE Computer Society, USA, 736–743. https://doi.org/10.1109/ICSE.2007.46

[21] Lilu Martin, Catherine Baker, Kristen Shinohara, and Yasmine N. Elglaly. 2022. The Landscape of Accessibility Skill Set in the Software Industry Positions. In *Proceedings of the 24th International ACM SIGACCESS Conference on Computers and Accessibility* (Athens, Greece) *(ASSETS '22)*. Association for Computing Machinery, New York, NY, USA, Article 65, 4 pages. https://doi.org/10.1145/3517428.3550389

[22] Michael Meng, Stephanie Steinhardt, and Andreas Schubert. 2019. How developers use API documentation: An observation study. *Communication Design Quarterly Review* 7, 2 (2019), 40–49.

[23] Rachel Menzies, Garreth W. Tigwell, Mandar Tamhane, and Annalu Waller. 2019. Weaving Accessibility Through an Undergraduate Degree. In *Proceedings of the 21st International ACM SIGACCESS Conference on Computers and Accessibility* (Pittsburgh, PA, USA) *(ASSETS '19)*. Association for Computing Machinery, New York, NY, USA, 526–529. https://doi.org/10.1145/3308561.3354611

[24] Meta. 2022. *React Native*. Meta, Palo Alto, CA. https://reactnative.dev/

[25] Kate Meyer. 2016. *Using the Microsoft desirability toolkit to test visual appeal*. Nielsen Norman Group. https://www.nngroup.com/articles/microsoft-desirability-toolkit/

[26] Maulishree Pandey, Sharvari Bondre, Sile O'Modhrain, and Steve Oney. 2022. Accessibility of UI Frameworks and Libraries for Programmers with Visual Impairments. In *2022 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. IEEE, New Jersey, NJ, USA, 1--10. https://doi.org/10.1109/VL/HCC53370.2022.9833098

[27] Maulishree Pandey, Vaishnav Kameswaran, Hrishikesh V Rao, Sile O'Modhrain, and Steve Oney. 2021. Understanding accessibility and collaboration in programming for people with visual impairments. *Proceedings of the ACM on Human-Computer Interaction* 5, CSCW1 (2021), 1–30.

[28] Rohan Patel, Pedro Breton, Catherine M. Baker, Yasmine N. El-Glaly, and Kristen Shinohara. 2020. Why Software is Not Accessible: Technology Professionals' Perspectives and Challenges *(CHI EA '20)*. Association for Computing Machinery, New York, NY, USA, 1–9. https://doi.org/10.1145/3334480.3383103

[29] Heydon Pickering. 2022. Inclusive Components - A blog trying to be a pattern library. All about designing inclusive web interfaces, piece by piece. https://inclusive-components.design/

[30] Athira Pillai, Kristen Shinohara, and Garreth W. Tigwell. 2022. Website Builders Still Contribute To Inaccessible Web Design. In *Proceedings of the 24th International ACM SIGACCESS Conference on Computers and Accessibility* (Athens, Greece) *(ASSETS '22)*. Association for Computing Machinery, New York, NY, USA, Article 71, 4 pages. https://doi.org/10.1145/3517428.3550368

[31] Cynthia Putnam, Maria Dahman, Emma Rose, Jinghui Cheng, and Glenn Bradford. 2016. Best practices for teaching accessibility in university classrooms: cultivating awareness, understanding, and appreciation for diverse users. *ACM Transactions on Accessible Computing (TACCESS)* 8, 4 (2016), 1–26.

[32] Cynthia Putnam, Kathryn Wozniak, Mary Jo Zefeldt, Jinghui Cheng, Morgan Caputo, and Carl Duffield. 2012. How Do Professionals Who Create Computing Technologies Consider Accessibility?. In *Proceedings of the 14th International ACM SIGACCESS Conference on Computers and Accessibility*

(Boulder, Colorado, USA) *(ASSETS '12)*. Association for Computing Machinery, New York, NY, USA, 87–94. https://doi.org/10.1145/2384916.2384932

[33] John T. Richards, Kyle Montague, and Vicki L. Hanson. 2012. Web Accessibility as a Side Effect. In *Proceedings of the 14th International ACM SIGACCESS Conference on Computers and Accessibility* (Boulder, Colorado, USA) *(ASSETS '12)*. Association for Computing Machinery, New York, NY, USA, 79–86. https://doi.org/10.1145/2384916.2384931

[34] Brian J. Rosmaita. 2006. Accessibility First! A New Approach to Web Design. In *Proceedings of the 37th SIGCSE Technical Symposium on Computer Science Education* (Houston, Texas, USA) *(SIGCSE '06)*. Association for Computing Machinery, New York, NY, USA, 270–274. https://doi.org/10.1145/1121341.1121426

[35] Anne Spencer Ross, Xiaoyi Zhang, James Fogarty, and Jacob O. Wobbrock. 2017. Epidemiology as a Framework for Large-Scale Mobile Application Accessibility Assessment. In *Proceedings of the 19th International ACM SIGACCESS Conference on Computers and Accessibility* (Baltimore, Maryland, USA) *(ASSETS '17)*. Association for Computing Machinery, New York, NY, USA, 2–11. https://doi.org/10.1145/3132525.3132547

[36] Anne Spencer Ross, Xiaoyi Zhang, James Fogarty, and Jacob O Wobbrock. 2020. An epidemiology-inspired large-scale analysis of android app accessibility. *ACM Transactions on Accessible Computing (TACCESS)* 13, 1 (2020), 1–36.

[37] Johnny Saldaña. 2015. *The coding manual for qualitative researchers*. Sage, London, England.

[38] Kristen Shinohara, Saba Kawas, Amy J. Ko, and Richard E. Ladner. 2018. Who Teaches Accessibility? A Survey of U.S. Computing Faculty. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education* (Baltimore, Maryland, USA) *(SIGCSE '18)*. Association for Computing Machinery, New York, NY, USA, 197–202. https://doi.org/10.1145/3159450.3159484

[39] Tailwind Labs Inc. 2023. *Tailwind UI*. Tailwind Labs Inc. https://tailwindui.com/

[40] The Apache Software Foundation. 2022. *Cordova*. Apache. https://cordova.apache.org/

[41] The National Network. 2023. *What is the Americans with Disabilities Act (ADA)? | ADA National Network*. The National Network. https://adata.org/learn-about-ada

[42] Shari Trewin, Shunguo Yan, Diane Margaretos, Michael Gower, Phill Jenkins, and Charu Pandhi. 2018. Feasibility of Automatically Assigning Severity Scores to Web Accessibility Problems. In *Proceedings of the 20th International ACM SIGACCESS Conference on Computers and Accessibility* (Galway, Ireland) *(ASSETS '18)*. Association for Computing Machinery, New York, NY, USA, 364–366. https://doi.org/10.1145/3234695.3241001

[43] Annalu Waller, Vicki L. Hanson, and David Sloan. 2009. Including Accessibility within and beyond Undergraduate Computing Courses. In *Proceedings of the 11th International ACM SIGACCESS Conference on Computers and Accessibility* (Pittsburgh, Pennsylvania, USA) *(Assets '09)*. Association for Computing Machinery, New York, NY, USA, 155–162. https://doi.org/10.1145/1639642.1639670