Stylette: Styling the Web with Natural Language

Tae Soo Kim
School of Computing, KAIST
Daejeon, Republic of Korea
taesoo.kim@kaist.ac.kr

DaEun Choi
School of Computing, KAIST
Daejeon, Republic of Korea
daeun.choi@kaist.ac.kr

Yoonseo Choi
School of Computing, KAIST
Daejeon, Republic of Korea
yoonseo.choi@kaist.ac.kr

Juho Kim
School of Computing, KAIST
Daejeon, Republic of Korea
juhokim@kaist.ac.kr

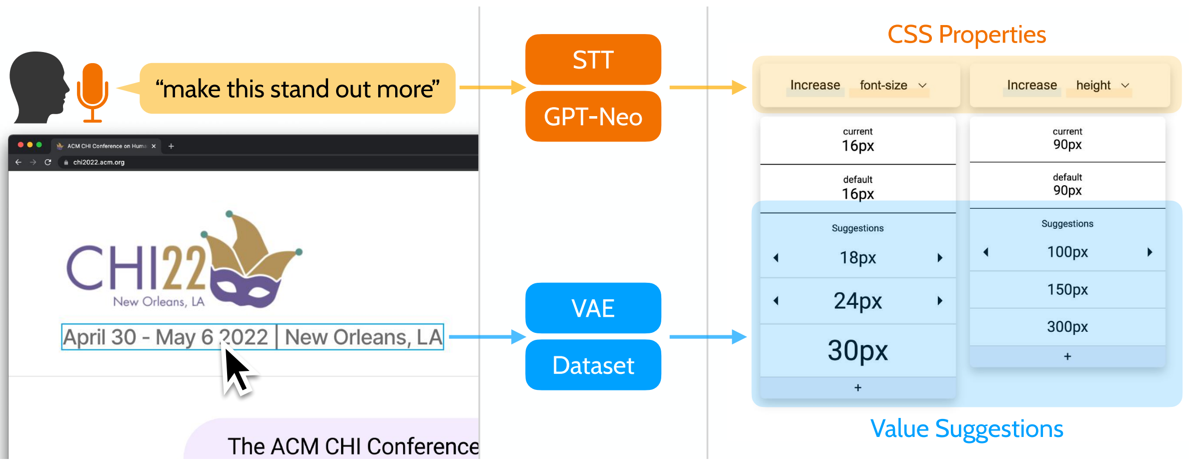


Figure 1: *Stylette* enables end-users to change the style of websites they visit by clicking on components and saying a desired change in natural language. A computational pipeline (1) transcribes the request and predicts plausible CSS properties with a large language model, and (2) encodes the clicked component using a convolutional neural network to identify and extract styling values from similar components in our large-scale dataset. These outputs are then presented in a *palette* that the user can use to iteratively change the component's style.

ABSTRACT

End-users can potentially style and customize websites by editing them through in-browser developer tools. Unfortunately, end-users lack the knowledge needed to translate high-level styling goals into low-level code edits. We present *Stylette*, a browser extension that enables users to change the style of websites by expressing goals in natural language. By interpreting the user's goal with a large language model and extracting suggestions from our dataset of 1.7 million web components, Stylette generates a *palette* of CSS properties and values that the user can apply to reach their goal. A comparative study (N=40) showed that Stylette lowered the learning curve, helping participants perform styling changes 35% faster than those using developer tools. By presenting various alternatives for a single goal, the tool helped participants familiarize themselves with CSS through experimentation. Beyond CSS, our work can be expanded to help novices quickly grasp complex software or programming languages.

CCS CONCEPTS

• **Human-centered computing** → **Natural language interfaces**; **Web-based interaction**; *Interactive systems and tools*; Empirical studies in HCI.

KEYWORDS

Web Design; Natural Language Interface; End-User Programming; Machine Learning

ACM Reference Format:
Tae Soo Kim, DaEun Choi, Yoonseo Choi, and Juho Kim. 2022. Stylette: Styling the Web with Natural Language. In *CHI Conference on Human Factors in Computing Systems (CHI '22), April 29-May 5, 2022, New Orleans, LA, USA.* ACM, New York, NY, USA, 17 pages. https://doi.org/10.1145/3491102.3501931

1 INTRODUCTION

The web is inherently malleable. Websites are rendered out of documents—HTML, CSS, and JavaScript code—which are transmitted to the user's browser and, thus, can be readily accessed and modified on the user side. This malleability allows users to improve their experiences on the web by personalizing pages [46], self-repairing existing issues [47], or even enhancing pages with additional features [28, 57, 63]. In addition, by sculpting others' creations, users can create their own new web pages [9, 39, 51]. The appeal of this malleability has led to the Greasemonkey [25] and Tampermonkey [5] plugins, which manage user scripts for these types of modifications, to collectively amass more than 10 million users. However, although such plugins allow users to install modifications designed by others, designing their own personal modifications may be out of reach for general end-users. To edit a web page's visual design or style, for example, users must be able to edit the underlying HTML and CSS files, but this requires an understanding of the code's language and structure. Thus, without the necessary expertise, most users are unable to mold websites into their own design.

To make the web more malleable for everyone, various end-user programming tools [34, 47, 58] have been designed to allow users with no expertise to directly manipulate a web page's visual design—abstracting away the underlying code. While these approaches allow users to focus on the visual representation, they require the user to manually perform several low-level operations (e.g., scrubbing on a color picker, typing in values) which can be tedious and effortful. Additionally, users must be able to decompose their high-level goals into the low-level operations supported by these tools—a task that inexperienced users frequently struggle with in other design-related tasks [1, 36]. Thus, to be able to easily transform a web page's design according to their goals, users require another level of abstraction.

Natural language interfaces allow users to perform complex, compound operations by simply saying or writing their intentions. The promise of this form of interaction has led to the development of various general-purpose voice assistants—e.g., Apple's Siri, Google Assistant, or Amazon's Alexa. In addition, task-specific natural language interfaces have also been designed to help inexperienced users perform complex tasks such as photo editing [36] or data visualization [24]. Similarly, if users could simply say what change they want to see, they could easily manipulate a web page without thinking about the underlying code or the low-level operations.

To investigate what language users would use when changing the style of a web page and how they would expect such changes to be presented, we conducted novice-expert sessions (N=8). In these sessions, novices used their voice to request changes on a web page's visual design and the expert, a developer, would then directly perform the changes using an in-browser developer tool. Our findings revealed that novices were frequently vague in their requests: omitting specific details (e.g., what color for the background), or using abstract terms that could not be clearly mapped to specific changes (e.g., "modern" or "vivid"). In addition to being vague due to inexperience, novices were also purposefully ambiguous as they wanted to explore the design space by seeing the expert's changes. Thus, novices expected the expert to make assumptions and provide a set of alternative changes that they could test and further iterate on.

Based on these findings, we designed Stylette, a natural language-based interface that assumes the user's intentions to provide a *palette* of web design properties and values. Stylette allows the user to modify a web component by clicking on it, and then saying or typing their desired change (e.g., "increase the size" or "make this cleaner"). Based on the user's input, the system provides a toolbox that contains (1) a set of CSS properties that could be changed to satisfy the request, and, (2) for each property, a set of alternative values to explore and sample. The user can then simply change the component by applying the different property values found in the toolbox. To generate these toolboxes, we designed a computational pipeline that processes and combines the two input modalities, natural language and clicks. Specifically, a GPT-Neo-based architecture predicts suitable CSS properties from the natural language request, and a variational autoencoder (VAE) model encodes the clicked-on component to extract the values of similar components from our dataset of 1.7 million components.

To evaluate Stylette, we conducted a between-subjects study (N=40) in which participants performed a design recreation task and an open-ended design task with either our system or DevTools, the Chrome Browser's developer tool. Our study revealed that Stylette helped participants perform styling changes 35% faster and with a higher success rate—80% of Stylette participants successfully recreated a design within the allowed time while only 35% succeeded with DevTools. Additionally, our system led participants to experiment with and familiarize themselves with a more diverse set of properties. As participants acquired more knowledge with Stylette, however, natural language interaction limited their productivity as they could not apply this knowledge to directly make changes themselves. These insights suggest a need for a hybrid approach: natural language interaction to initially support quick familiarization with a tool, and then gradually phasing in more direct interaction methods.

This paper presents the following contributions:

(1) Stylette: A novel system that allows users to change the design of websites by using natural language to express their goal, and then iterating with the set of alternatives presented by the system.
(2) A computational pipeline that combines NLP and CV techniques to process a natural language request and a web component into a set of plausible CSS property and value changes.
(3) Findings from a between-subjects study that reveals how natural language support can help novices familiarize with and perform a previously unknown design/coding task.

2 RELATED WORK

We aim to enable end-users to modify the design of any website by simply saying what change they need. To this end, we review related work in (1) web manipulation tools, and (2) designing and (3) coding with natural language.

2.1 Web Design and Manipulation Tools

As web interfaces are visual representations of HTML and CSS code, various tools have been designed to facilitate the process of modifying the code to produce desired visual changes. For example, openHTML [49] provides an educational environment which shows HTML code, CSS code, and a website preview side-by-side. Other systems [9, 39, 51] allow users to inspect the code behind pages to understand the connection between code and visuals. Beyond inspection, Chickenfoot [7] allows end-users to write simple scripts to modify components, and, more recently, Spacewalker [67] leverages genetic algorithms and crowdsourcing to generate design alternatives. These tools, however, were designed with developers or learners in mind, and require the user to understand and interact with the code—a task impractical for end-users with limited knowledge.

To make manipulation more practical, a separate line of research allows users to modify a website's visuals by directly interacting with the visuals. CrowdAdapt [47], CrowdUI [48], and XDBrowser [46] allow users to modify the positioning of web components through direct manipulation (e.g., drag-and-drop). Aimed at designers who have limited coding knowledge, Poirot [58] and CoCapture [13] provide designer-specific widgets to support design editing and animation authoring, respectively, directly on websites. These tools, however, still require the user to expend time and effort deciding between and performing various possible editing operations. Example-based systems [16, 33, 37] aim to reduce this mental and manual effort by allowing users to copy the styles of other websites. Our work aims to simplify the process further: modifying a website's design by simply describing a change and selecting from suggested alternatives—without deciding on operations or looking for examples.

2.2 Designing with Natural Language

Novices struggle to translate high-level design goals into tool operations due to the vocabulary problem [22]—the language used by the user and the tool do not match. Thus, empowering users to be able to design by simply stating their high-level goals has been a long-standing goal for HCI researchers. Query-Feature Graphs (QF-Graphs) [18] and CommandSpace [1] jointly modeled natural language descriptions with feature names in design applications (e.g., GIMP and Photoshop) to help users identify features based on their needs. Other systems support the use of natural language concepts to search for design references or components—images [17], graphic designs [29], or 3D models [10]. Beyond searching, several systems generate design artifacts (e.g., images [35] or icons [65]) based on the semantic meaning of words. Leveraging whole expressions instead of only words, Crosspower [60] and PixelTone [36] decompose expressions into operations for animation authoring and image editing, respectively. Our work expands this line of research by interpreting vague natural language expressions, which cannot be clearly decomposed, to support design editing in the context of web pages.

2.3 Coding with Natural Language

A crucial step in programming is coding—writing instructions in the form of machine-readable syntax. To lower the barrier to coding, substantial effort has been dedicated to bridge natural language and complex programming languages [45]. For instance, researchers have used semantic parsers [50] and bimodal models [2] to map natural language to code. Such techniques enabled systems that allow novice coders to quickly search for code snippets [52, 53], and non-coders to code small programs by demonstrating and describing tasks [38, 44]. Beyond mapping, a line of work has also developed techniques that take natural language as input and generate code—e.g., Python [41, 61], Bash commands [40], SQL queries [68], or API calls [59]. Recent advancements in natural language processing (NLP), and especially in large language models (e.g., GPT-3 [8]), have led to performance boosts in natural language-based code generation [27]. OpenAI's Codex [12], a GPT-3-based model, is able to generate basic games from a few natural language sentences [62]. In this same line of research, our work leverages a large language model to facilitate editing of CSS code and provides insight into how natural language interaction can scaffold novices' learning of coding languages.

3 FORMATIVE STUDY

We conducted a formative study to investigate how novices would change the design of websites and how they would naturally request such changes. In this study, participants freely browsed through a website and requested styling changes by speaking aloud. One of the researchers, with several years of development experience, acted as an expert and made these changes on-the-go.

3.1 Participants

We invited 8 participants (5 female, 3 male), all of whom had no background in web development. Each participant sat alongside the expert or, if participating remotely, shared their screen through a video conferencing tool[1]. To reduce the time participants spent familiarizing themselves with a website and to prompt more realistic requests, participants chose a website they frequently visit for the study. Most participants chose either our university's web portal or its learning management system.

3.2 Study Procedure

During the study, participants were asked to examine the website, and request any styling changes that they want or could improve their future experiences on the site. On their own computer, the expert used the Chrome Browser's DevTools[2] to directly edit the CSS code. The expert would then share the edits and, if participants were not satisfied, they could ask for further edits. After around 30 minutes of editing, the participants were then asked a couple of questions about their experience. Sessions lasted a maximum of 40 minutes and participants made 8.38 requests on average.

3.3 Requests were Vague and Abstract

Despite being able to concretely specify which web component they wanted to edit, participants struggled to concretely explain how it should be changed. Participants generally relied on vague phrases (e.g., "more readable" or "emphasize this") or abstract terms (e.g., "modern", "vivid" or "dull") that did not immediately reveal what

[1]https://zoom.us
[2]https://developer.chrome.com/docs/devtools/

visual aspect of the component should be changed or how. Even if they were specific about which aspect to change, participants would also tend to be vague about the value to set for that aspect. For instance, a participant said "more transparent" without specifying how much more transparent.

We observed that the behavior of our participants was beyond not knowing the names of CSS properties—like the vocabulary problem observed in other tasks [22]. Participants also struggled to specify the visual aspects of the web components even without using the actual property names. For example, a participant requested a text component to be highlighted but, when asked if the text should be bolder or colored differently, they were unable to provide a definite answer. Participants explained that their hesitation was either because (1) they were unsure about which aspect to change, or (2) they could decide on an aspect but were not confident that it would "look good".

3.4 Assumptions Over Questions

To concretize the participants' vague requests, the expert asked questions to prompt further details. For example, when a participant asked to make a component "less tacky", the expert asked about what made it appear "tacky". While participants recognized how these questions helped them iteratively decompose their goals, they found this back-and-forth to be tedious. As participants were unsure about the details, they did not want to dedicate the mental effort to ponder about the details and, instead, expected the expert to assume the details for them. They mentioned that it would be easier to distinguish what they liked or disliked if the expert made these assumptions and presented a visual result. Additionally, instead of one outcome for each request, participants wanted various options for the same request in order to explore the design space.

3.5 Natural Language is Not a Panacea

For most participants, the use of voice or natural language was a major positive aspect about interacting with the expert. Participants mentioned how it was "comfortable" to use natural language to simply explain what they wanted to change. However, while they felt that natural language helped to get the editing process started, participants desired more direct control when iterating on edits. Specifically, when deciding on a value for a property, they felt frustrated about having to test different values by turn-taking with the expert. Instead, participants wanted to be presented with widgets that allowed them to test alternative values by themselves.

Based on the study insights, we derive the following design goals:

- DG1: Interpret vague requests to present plausible changes.
- DG2: Provide multiple alternative properties and values that could satisfy one request.
- DG3: Allow users to directly iterate on the details for a change.

4 STYLETTE

Based on our design goals, we present Stylette (Fig. 2), a system that enables end-users to change the visual design of any website by simply clicking on a component and saying what change they want to see. The system interprets the user's request through an NLP pipeline trained on vague language (DG1) to present a *palette* that consists of multiple CSS properties (DG2) that could be changed to satisfy the request. To iteratively edit each property, the user can directly adjust values and experiment with various suggestions extracted from a large-scale dataset (DG2, DG3). Stylette is implemented as a Chrome Extension and, using a method similar to Tanner et al.'s [58], it saves the user's changes in the browser's memory so that they persist when the user returns to the page.

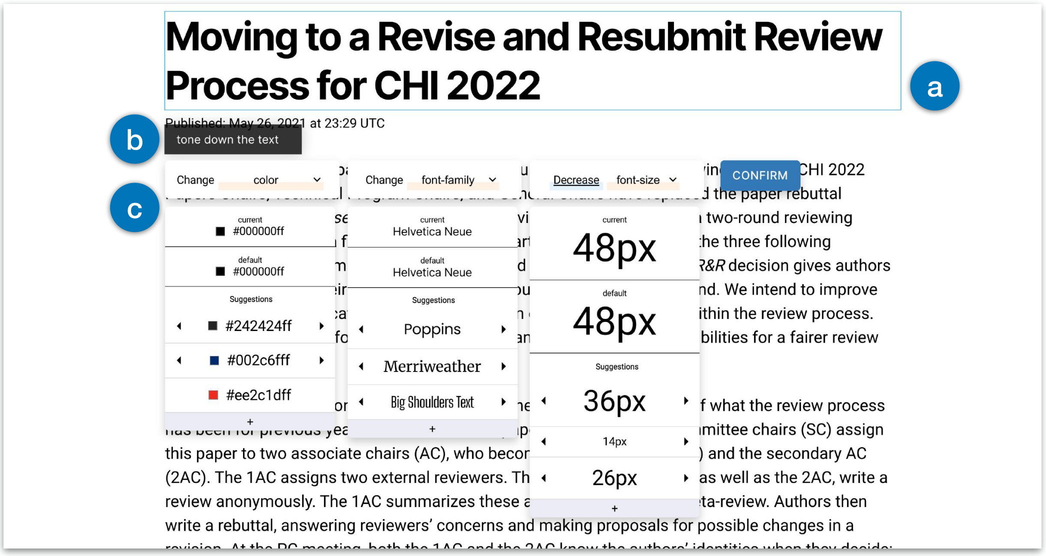


Figure 2: *Stylette* is shown overlaid on a website. When activated, the system shows a blue border (a) over components the user has hovered-on or clicked. After the user selects a component and records a request, Stylette transcribes the request (b) and displays a *palette* that contains CSS properties and values.

4.1 User Scenario

To illustrate how Stylette can be used, we follow Sofia, a sociologist preparing for a paper submission to CHI 2022. While Sofia frequently visits the conference's website to check for submission details or recent news, she feels that the design can make it challenging to look for and read the contained information. As she has no web development experience, she decides to use Stylette to make styling changes to the website.

4.1.1 Selecting a Component. In the frontpage of the CHI website, Sofia feels that the header text is overemphasized and prevents several news posts from being seen in one glance. To start editing, she clicks on the Stylette icon on her extension toolbar. Now, she can select components to edit so she clicks on the first header in the page (Fig. 2a).

4.1.2 Making a Verbal Request. With the component selected, Stylette overlays a transcript box on the website, prompting Sofia to say her request. To do so, she holds down the **Ctrl** key and says: *"tone down the text"* (Fig. 2b). After releasing the **Ctrl** key and a short processing period, the transcript box now displays a transcript of what Sofia said. In case the transcription is wrong, Sofia can correct it by typing directly on the box and pressing **Enter** to process the corrected transcript. In addition, a *palette* is now presented, showing three different CSS properties that Sofia can edit to satisfy her needs (Fig. 2c): she can make the text smaller with *font-size*, change it to a slimmer *font-family*, or apply a lighter *color*.

4.1.3 Iterating with the palette. Under each property, the *palette* presents a list of values: the current value for the property (Fig. 3a), the default or original value (Fig. 3b), and a set of value suggestions (Fig. 3c). For properties with numerical values, like *font-size*, the system also interprets whether the user wants to increase or decrease the current value (Fig. 3d) and provides suggestions accordingly. As Sofia hovers over the suggested values for *font-size*, Sofia can see how the header would look with that *font-size*. After finding one she feels satisfied with, she clicks on it to apply that change. If Sofia actually wanted to increase the *font-size* and the system gave an incorrect prediction, she could click on "Decrease" next to the property name to switch it to "Increase" and the suggestions would change accordingly. If she wanted to change another property similar to *font-size*, she could also click on the property name to see a drop-down of other properties with similar names (e.g., *font-style*, *font-weight*).

After setting the *font-size*, Sofia also notices the *color* property. As she feels that this could also be toned down a bit, she clicks on the lighter black color ("#242424ff") in the suggestions. After seeing this change, she feels that the header's color should be even lighter, so she clicks on the arrows next to that suggestion (Fig. 3e) to see other similar suggestions. Going through the carousel, she finds a color that she likes so she clicks on it. If she is unsatisfied with the suggestions, she can see other different suggestions by clicking on the "+" at the bottom (Fig. 3f), or manually set her own value by clicking on the current value to expose manual change widgets (Fig. 3g).

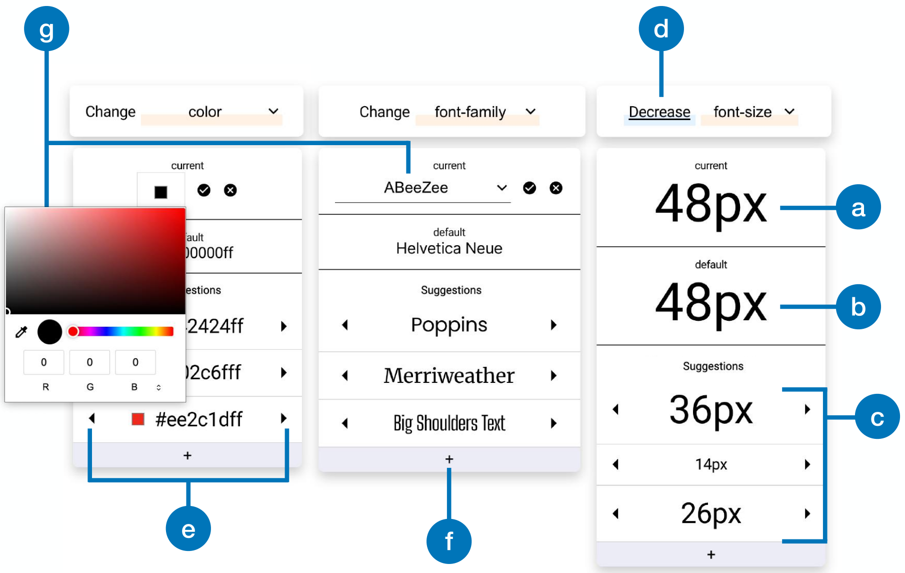


Figure 3: For each property, the *palette* presents the current value (a), the default or original value before any changes (b), and a list of suggested values (c). For numerical values, the palette presents suggested values that are either larger or smaller than the current value based on the system's prediction (d). To see other similar suggestions, the user can click on the arrows next to a suggested value (e). To see different suggestions, the user can click on the "+" button (f). The user can also click on the current value to reveal widgets to manually set values (g): input box for numerical properties (e.g., *font-size*), drop-down menu for nominal properties (e.g., *font-family*), or color picker for colors.

4.2 Pipeline

To support the interaction presented in the scenario, we present a computational pipeline that processes the two input modalities, voice and click, to generate the *palettes* (Fig. 4). For voice, the audio is recorded and automatically transcribed. For clicks, a screenshot of the component selected by the user is automatically captured. These inputs are then processed separately by the computational pipeline.

4.2.1 Processing Natural Language. Our pipeline's NLP module takes the transcribed request, and predicts relevant CSS properties and the direction of the change (e.g., increase, decrease, or neither). For this purpose, we employ the 2.7 billion parameter version of the GPT-Neo model [6], an open-source implementation of OpenAI's GPT-3 model [8]. With a well-crafted prompt and a small number of examples, these models have been shown to achieve high performance on previously unseen tasks. However, hand-crafting prompts can be a time-consuming and very imprecise process—small alterations can lead to significant differences in performance.

Architecture: Instead, we implement the P-tuning technique [42] that automatically searches for a prompt with high performance. In this technique, prompts are composed by concatenating *pseudo-tokens* to the natural language input and training the embeddings for these pseudo-tokens. In our pipeline, we use 12 pseudo-tokens. For training, we template the prompt as [$P_{1:4}$,R,$P_{5:8}$,C,$P_{9:12}$,D], where $p_{1:12}$ are the pseudo-tokens, R is the natural language request, C is the CSS properties separated by commas, and D is the change direction (i.e., "increase", "decrease", or "none"). During inference, we template the prompt as shown in Figure 4 ("Concatenate"): [$P_{1:4}$,R,$P_{5:8}$]. This templated prompt is passed as input to the model and the model's output is controlled to generate at least three CSS properties—to provide multiple alternatives to users ("Input" and "Generate" in Fig. 4). Then, the generated CSS properties and the remaining pseudo-tokens are concatenated to the initial prompt, and this result is passed to the model again to generate the change direction.

Dataset: Another merit of the P-tuning technique is that it only requires a small amount of data for training. To train our pseudo-tokens, we created a small-scale dataset consisting of 300 triplets of (1) vague natural language requests, (2) CSS property sets, and (3) change directions. As a first step in creating this dataset, we requested 29 web developers to each write three hypothetical vague requests that a user could ask when wanting to change a website's design. Then, each developer looked at the requests written by another person and wrote CSS properties to change and the direction for the change that could satisfy each request. We removed requests that were too specific (e.g., included property names), and added requests from our formative study and system's pilot studies. The CSS properties in this initial data are the ones supported in our system (Table 2). We then expanded the dataset by automatically augmenting the initial data with synonym/antonym replacement [43, 64], and/or backtranslation [54]—one of the authors checked and corrected the augmentations. Finally, as performance can deteriorate

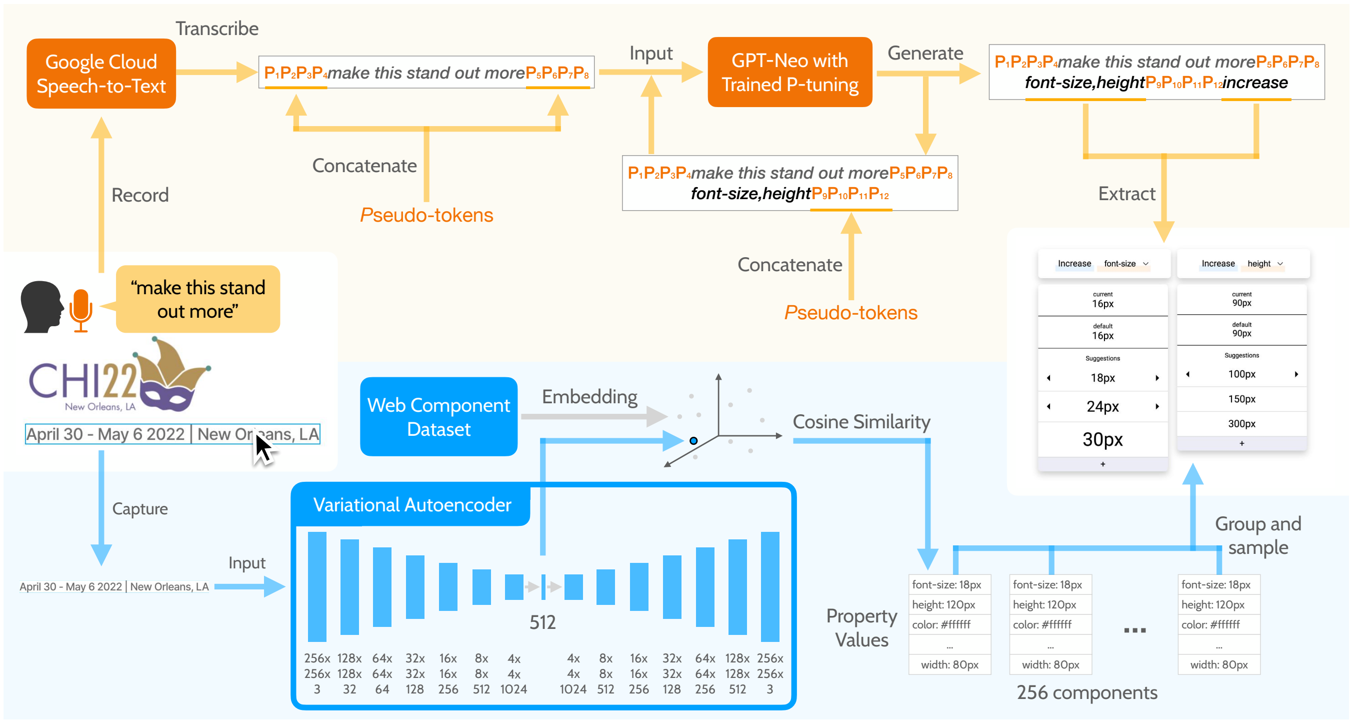


Figure 4: Our computational pipeline integrates a natural language processing (NLP) module (top, orange) and a computer vision (CV) module (bottom, blue). The diagram illustrates the pipeline at inference time—processing user's input of natural language and clicks to generate a set of CSS property alternatives and value suggestions.

Model	CSS Property Prediction				Direction
	Acc.	Pre.	Rec.	F1	Acc.
P-tuning	0.557	0.670	0.761	0.653	0.819
Hand-crafted	0.509	0.623	0.648	0.585	0.413

Table 1: With trained P-tuning, the GPT-Neo model achieved higher performance when predicting CSS properties and change directions, when compared to using a hand-crafted prompt as input.

significantly due to class imbalance [66], we ensured that each CSS property appeared in at least 10% of the requests—the representation of CSS properties in the dataset is shown in Table 2. After augmentation and balancing, we finalized our dataset of 300 triplets.

Training: In the training process, we used 200 triplets for training and validation (80%-20% split), and reserved 100 for testing. The pseudo-tokens were trained on the generative loss from the GPT-Neo model with the Adam optimizer, until early stopping on the validation loss. We used an initial learning rate of 0.0001, batch size of 8, weight decay value of 3e-7, and gradient clipping value of 5. When compared to the model with our best hand-crafted prompt, GPT-Neo with trained P-tuning achieved a higher F1-score when predicting CSS properties and higher accuracy when predicting change direction (Table 1). Additionally, the recall with P-tuning exceeds 75% which suggests that, for the average request, the model will likely return most of the properties that the user might need.

4.2.2 Processing Web Components. As our formative study revealed, users can struggle when deciding on a value for a change (e.g., what color for the background) and may benefit from seeing various alternatives. The components in other websites can be a rich source for these alternatives. However, as the style of a component depends on what that component represents (e.g., the *font-size* for a header vs that for a paragraph), selecting random components would not lead to sensible and useful alternatives. Thus, it would be more beneficial to identify components in other websites that are similar to the one the user wants to change. Similarity could be measured by calculating property differences and aggregating these into one measure, but, as properties differ in the scale and type of values, this requires the difference and aggregation calculations to be carefully formulated.

Architecture: As an alternative, we leverage a variational autoencoder (VAE) model [30] ("Variational Autoencoder" in Fig. 4) to automatically learn a concise representation of the visual features of components. In our pipeline, we use this VAE model to encode the screenshot image of a component into a 512-dimensional vector. Through cosine similarity, this vector is then compared to the vector representations of all the components in our large-scale dataset to identify 256 similar components and retrieve their property values ("Cosine Similarity" in Fig. 4). To provide coarse diversity but also more fine-grained alternatives, the *palette* presents suggested values in two levels: (1) different values as separate rows, and (2) similar values as a carousel in the same row. To support this, the pipeline groups the retrieved values according to specific rules ("Grouping Method" in Table 2) and each group represents a suggestion row. Then, a maximum of 10 values are randomly sampled for each group and these alternatives are presented through the carousel. For color-related properties and the *font-family* property, users in the pilot studies wanted more diverse values so, for these properties, we populate other suggestion groups by retrieving the values from random components in the dataset.

CSS Property	Grouping Method	Percent
height	Interval binning (N=20)	11.7%
width	Interval binning (N=20)	11.7%
margin	Interval binning (N=10)	11.3%
padding	Interval binning (N=10)	12.9%
color	K-means clustering (N=6)	12.9%
background-color	K-means clustering (N=6)	12.5%
opacity	Interval binning (N=2)	10.4%
font-size	Interval binning (N=10)	13.3%
font-family	Google Fonts categories (N=5)	13.8%
font-style	Nominal value	11.3%
font-weight	Interval binning (N=10)	11.7%
text-align	Nominal value	11.3%
text-decoration	Nominal value	11.3%
border-width	Interval binning (N=10)	11.3%
border-color	K-means clustering (N=6)	11.3%
border-radius	Interval binning (N=10)	11.3%

Table 2: The CSS properties supported by Stylette. The table presents the representation of each property in the natural language request dataset as a percentage. Each row also shows how values for a property are grouped when suggested to the user: interval binning into N equally-spaced intervals, K-means clustering with the elbow method, based on the categories from Google Fonts, and no grouping for properties with nominal values.

Dataset: Although there are datasets for mobile UI components [11] or for whole web pages [32], there are none for individual web components. Thus, to train the VAE model, we constructed our own dataset. We first compiled a list of websites from various sources: the S&P500, the Webby Awards [4], and the Open PageRank dataset [15]. We removed any websites that (1) could not be accessed, (2) had very similar URLs, or (3) had less than 16 components. This led to a final list of 7,565 websites. For each website's main page, we used a crawler to capture each component's CSS properties and screenshot image. After removing components that were less than 10 pixels wide or tall, the final dataset consisted of 1,761,161 components.

Training: The VAE model is composed of six convolutional layers for encoding, one linear layer as a bottleneck, and six transposed convolutional layers for decoding. The dimensions of the outputs at each layer are shown in Figure 4 ("Variational Autoencoder"). During training, the image of a component is encoded into a vector using the encoding and bottleneck layers, and then this vector is passed through the decoding layers to recreate the image. The model is trained to maximize the evidence lower bound (ELBO) value between the original image and the recreated image. We trained our model for 3 epochs with an Adam optimizer, using a learning rate of 0.0001 and batch size of 256.

4.3 Implementation

We implemented the interface of Stylette as a Chrome Extension, using JavaScript, HTML, and CSS. For the backend, we used a Node.js server to pre-process requests from the interface and transcribe the audio with the Google Cloud Speech-to-Text API[3]. To serve the computational pipeline, we used a Flask server running with DeepSpeed[4].

5 EVALUATION

We conducted a between-subjects study where we compared Stylette against the Chrome Browser's DevTools, a tool widely available for general end-users to edit websites with. The study was composed of (1) a well-defined task of redesigning a website to look like a given outcome, and (2) an open-ended task of styling a website to follow the design direction of provided references. We designed these two tasks to investigate how Stylette helped participants style components when (1) they have a clear idea about how it should change, or (2) they only have a vague sense of direction. Specifically, we pose the following research questions:

- RQ1. How does Stylette help novice users find the CSS properties required to perform desired styling changes?
- RQ2. Can Stylette encourage novices to perform a greater number of changes and use more diverse CSS properties?
- RQ3. How does novices' usage of Stylette affect their self-confidence regarding their own web designing abilities?

5.1 Participants and Apparatus

We recruited 40 participants (11 female, 29 male; age M=21.5 and SD=3.05) who all reported having no previous experience with web design or coding (no knowledge of HTML and CSS). We also verified that participants were relatively fluent in spoken English to reduce frustration due to the performance of speech-to-text technologies. Participants were divided into two equally-sized groups and each group was assigned to use either Stylette or Chrome DevTools. As six participants mentioned having other prior design experiences and this could affect performance (e.g., the term "padding" is used in other design tasks), they were also equally split into each condition. To simulate a realistic setting, participants who used Chrome DevTools were also allowed to freely use search engines to find resources and information. The study lasted a maximum of 90 minutes and participants were compensated with 30,000 KRW (approximately 26 USD).

5.2 Study Procedure

The study took place face-to-face, strictly following the COVID-19 guidelines: participants had to wear masks and plastic gloves, and their temperature was checked before sessions. Each participant was provided with a computer with a Chrome browser installed and their assigned tool, Stylette or DevTools, already opened. After reading and signing the informed consent form, participants were first provided with a brief walkthrough of their assigned tool and were then allowed to test the tool for a total of 5 minutes. After this, participants completed a short pre-task survey.

[3]https://cloud.google.com/speech-to-text

[4]https://www.deepspeed.ai/

Tag	Properties to Change	Success Range
h2	font-size (FSz)	80px - 120px
p	font-weight (FW)	700 - 900
span	background-color (BgC)	(0.0, 0.0, 0.6) - (0.2, 0.2, 1.0)
	color (C)	(1.0, 1.0, 1.0) - (0.8, 0.8, 0.8)
video	border-radius (BR)	60px - 100px
button	border-width (BW)	6px - 10px
	border-color (BC)	(0.8, 0.4, 0.0) - (1.0, 0.8, 0.2)
div	text-align (TA)	"center"
h2	font-style (FSt)	"italic" or "oblique"
button	padding (P)	30px - 60px
img	width (W)	600px - 800px
h3	font-family (F)	Any in "cursive" category
h3	text-decoration (TD)	"underline"
div	margin (M)	80px - 120px
img	height (H)	350px - 450px
img	opacity (O)	0.3 - 0.7

Table 3: List of the components that participants had to change during Task 1, in the order that they had to be changed. For each component, the table shows its tag type and the properties that had to be changed. For the properties, the list also shows their abbreviated names (which are used hereafter), and the range of values that were accepted as successful changes (color values shown as RGB triplets).

After the survey, participants started Task 1. Participants were tasked with using their assigned tool to redesign our institute's "About" web page[5] to look as close as possible to a provided final design. This final design was provided as a before-after image with circling around components to change and labels showing how many properties to change for each component. Natural language explanations of the changes were not provided to prevent biasing the language used by Stylette participants. The task involved changing 14 different components and all of the 16 CSS properties supported by our system (Table 3). Participants were asked to change the components in the order that they appeared in the website, but were allowed to skip challenging components and come back to them later. A researcher verified that a component had been successfully changed once the values of the correct properties were within the accepted success range ("Success Range" in Table 3). Participants had 30 minutes to successfully change all the components. After the task, participants completed a short survey.

After Task 1, participants started Task 2 after a 5-minute break. To ensure that all participants started Task 2 with the same amount of knowledge, those that did not complete Task 1 were first shown how to perform the changes that they did not complete. The aim of Task 2 was to investigate how participants used their assigned tool when only provided with a vague direction for changes and allowed greater flexibility. Participants were tasked with changing a given website such that it followed the design direction of four reference websites (Fig. 5). These references were chosen as they shared a similar modern aesthetic, but also differed in how their content was structured. Participants were given 25 minutes for this

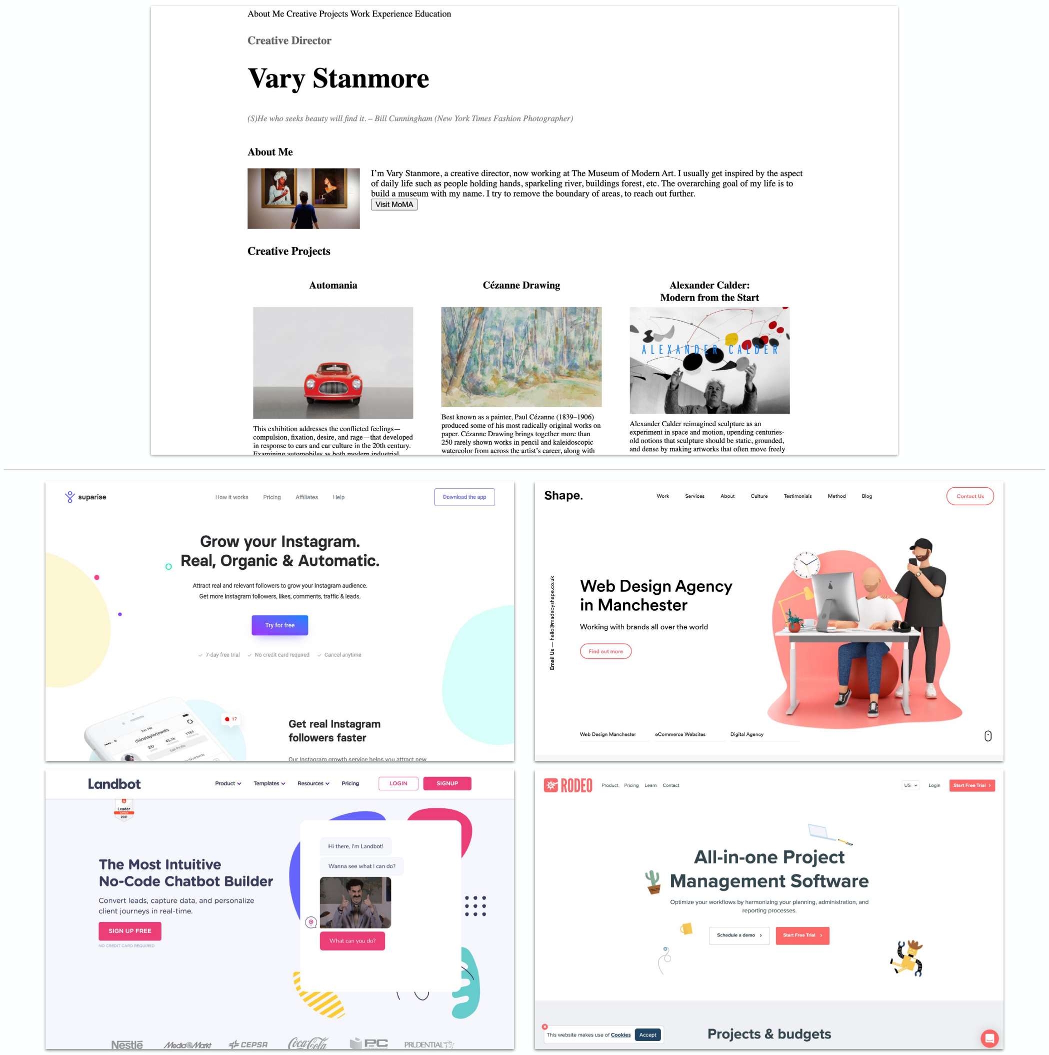
[5]https://www.kaist.ac.kr/en/html/kaist/01.html

task. After this task, participants completed a short survey. Finally, a short interview was conducted asking participants about their experiences during both tasks.

5.3 Measures

We collected responses to the pre-survey and the post-surveys after each task. All of the surveys contained four questions asking participants to rate, on a 7-point Likert scale, their self-confidence with respect to their ability to (1) perform a website design changing task, (2) plan design changes, (3) iterate on changes, and (4) use the given tool. We averaged the responses to these questions to derive one score for self-confidence. The two post-surveys included the six questions from the NASA-TLX questionnaire [26] to measure participants' perceived workload.

We also quantitatively measured task-related metrics. For Task 1, we measured the time taken to successfully change each component and to complete the whole task. We hypothesized that Stylette would help participants find properties faster, and therefore complete changes in less time. Although all participants had no previous web design experiences and those with other design experiences were equally divided into each condition, individual design interest and skill could still affect the quality of the final designs in Task 2. Due to this reason, we did not rate these designs and, instead, we measured how many property changes were made in total. We hypothesized that Stylette participants would make more changes as they could explore diverse properties and values. A value close to 0 indicates equal usage, spread across various properties, and one close to 1 indicates unequal usage, few properties used excessively.



About Me Creative Projects Work Experience Education

Creative Director

Vary Stanmore

(S)He who seeks beauty will find it. – Bill Cunningham (New York Times Fashion Photographer)

About Me

I'm Vary Stanmore, a creative director, now working at The Museum of Modern Art. I usually get inspired by the aspect of daily life such as people holding hands, sparkeling river, buildings forest, etc. The overarching goal of my life is to build a museum with my name. I try to remove the boundary of areas, to reach out further.

Visit MoMA

Creative Projects

Automania

This exhibition addresses the conflicted feelings—compulsion, fixation, desire, and rage—that developed in response to cars and car culture in the 20th century.

Cézanne Drawing

Best known as a painter, Paul Cézanne (1839–1906) produced some of his most radically original works on paper. Cézanne Drawing brings together more than 250 rarely shown works in pencil and kaleidoscopic watercolor from across the artist's career, along with

Alexander Calder: Modern from the Start

Alexander Calder reimagined sculpture as an experiment in space and motion, upending centuries-old notions that sculpture should be static, grounded, and dense by making artworks that often move freely

suparise
How it works
Pricing
Affiliates
Help
Download the app
Grow your Instagram.
Real, Organic & Automatic.
Attract real and relevant followers to grow your Instagram audience.
Get more Instagram followers, likes, comments, traffic & leads.
Try for free
Get real Instagram followers faster
Shape.
Work
Services
About
Culture
Testimonials
Method
Blog
Contact Us
Web Design Agency in Manchester
Working with brands all over the world
Find out more
Landbot
LOGIN
SIGNUP
The Most Intuitive No-Code Chatbot Builder
Convert leads, capture data, and personalize client journeys in real-time.
SIGN UP FREE
Hi there, I'm Landbot!
What can you do?
RODEO
All-in-one Project Management Software
Schedule a demo
Start Free Trial
Projects & budgets

Figure 5: (Top) The website that participants styled in Task 2 mimics the portfolio of a creative director for a museum. The website only has basic styling to encourage participants to be creative and make many changes. (Bottom) The four reference websites that were provided during the task: Suparise (https://suparise.com), MadeByShape (https://madebyshape.co.uk), Landbot (https://landbot.io), and Rodeo (https://getrodeo.io).

For qualitative data, we analyzed participants' responses during the short interviews to understand their perceptions of the given tool and how they leveraged it for their purposes. We also iteratively coded the requests used by Stylette participants in Task 2 to classify them according to the vagueness of their content and language.

6 RESULTS

Our results demonstrated that Stylette helped participants perform styling changes faster and with greater success in Task 1, but it did not enhance productivity in Task 2. For the statistic analysis of each measure, we first conducted a Shapiro-Wilk test to determine if the data was parametric (noted with "P") or non-parametric (noted with "NP"). When comparing between conditions, we used an independent t-test (if parametric) and a Mann-Whitney U test (if non-parametric). When comparing between tasks within the same condition, we used a paired t-test (if parametric) and a Wilcoxon signed-rank test (if non-parametric).

6.1 Task 1: Well-Defined Task

To answer RQ1, we analyzed participants' performance in Task 1 (i.e., time taken and success rate to perform the given changes). We additionally measured participants' perceived workload.

6.1.1 Performance. Overall, Stylette participants significantly outperformed DevTools participants in this task (Fig. 6). While only 7 out of 20 DevTools participants completed all changes, 16 out of 20 Stylette participants completed the task. Additionally, when comparing only those who completed the task, Stylette participants (M=971.8s, SD=314.4s) completed the task in 35% less time than those that used DevTools (M=1493.0s, SD=295.9s, t=-3.72, p=0.001, P). Comparing the time taken to successfully change each component revealed that DevTools participants struggled significantly with specific properties (e.g., *border-radius* (BR) and *padding* (P) in Fig. 6). These struggles generally involved two scenarios: (1) vague search queries led to unhelpful results, or (2) the name of a CSS property did not immediately reveal its visual function.

To illustrate the first scenario, several participants tried queries like *"enlarge the border in CSS"* when searching for the *padding* property, but this only returned results for *border-width*—the search engine took them "too literally" (D2, D7, D9, D14). In other cases, participants' vague queries returned search results for more advanced changes beyond their needs. For example, to adjust the *height* or *width*, participants searched "resize image in CSS" but this returned results about "responsive images". In contrast, as our system was trained on vague requests and presents multiple properties for one request, Stylette participants had more success using similarly vague language—requesting *"enlarge the border"* to the system returned *padding* among the options.

The second scenario involved properties with names that could be unclear for novices, such as *border-radius* or *text-decoration.* In these situations, the properties were frequently found in the search results, but DevTools participants would overlook them as they could not immediately visualize the functions from the names or mismatched with their mental models. However, hovering over the suggested values allowed them to quickly use and test the functions of properties. S5 mentioned: *"By applying [the recommendations], I could understand what [visual] concept the [margin and padding] were related to".*

6.1.2 Perceived Workload. In Task 1, responses to the NASA-TLX questions revealed that the effect of Stylette on perceived workload was mixed (Table 4). Stylette participants reported experiencing significantly less temporal demand (U=118.0, p=0.0118, NP) and effort (U=133.0, p=0.0336, NP) than those that used DevTools. DevTools

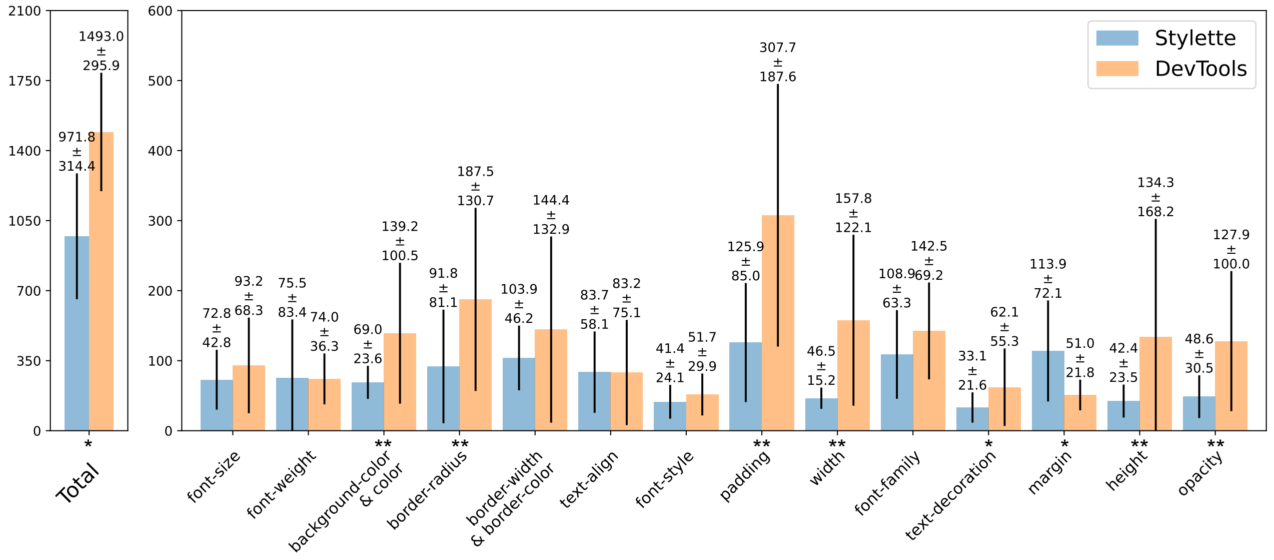


Figure 6: The average time taken for participants to successfully change each component using Stylette or DevTools. Each component is represented with the abbreviated names of the properties changed (Table 3). For each property, the figure shows if the difference in time taken for each condition was statistically significant (*: p <.05, **: p < .01).

Task	Condition	Mental	Physical	Temporal	Effort	Performance	Frustration
1	**Stylette**	3.90 (1.41)	2.55 (1.57)	3.45 (1.73)	3.15 (1.79)	5.45 (1.10)	3.00 (1.52)
	DevTools	4.35 (1.42)	1.65 (0.93)	4.50 (0.95)	4.00 (1.45)	4.85 (1.53)	2.25 (1.21)
	p	0.14	**0.02**	**0.01**	**0.03**	0.11	**0.05**
2	**Stylette**	4.75 (1.29)	2.90 (1.71)	4.35 (1.66)	4.25 (1.48)	4.05 (1.43)	3.55 (1.47)
	DevTools	4.90 (1.21)	2.00 (1.45)	4.55 (1.50)	4.55 (0.83)	4.45 (1.39)	2.65 (1.50)
	p	0.34	**0.03**	0.69	0.23	0.24	**0.03**

Table 4: For Task 1, participants' average ratings on the perceived workload questions (NASA-TLX) showed that temporal demand and effort were significantly lower with Stylette, but physical demand and frustration were significantly higher. For Task 2, physical demand and frustration were still rated significantly higher with Stylette, but temporal demand and effort no longer differed significantly.

participants felt significant time pressure due to the lengthy and effortful process of thinking about what to search, skimming through search results, and reading resources. In comparison, Stylette participants could simply say something and look through the three to five properties presented by the system.

However, Stylette participants also experienced significantly higher frustration when compared to DevTools participants (U=139.5, p=0.0472, NP). According to participants, this frustration was partially attributed to the fact that the coupled AI algorithms (i.e., speech-to-text and property prediction) could both fail. For example, as they did not notice the transcription errors, several participants were confused when concrete requests (e.g., "underline text") did not return the correct properties. Other participants were overly preoccupied with the transcription and immediately corrected any errors—failing to notice that the system had already returned desired properties. When fixing errors, participants also had to alternate between modalities (i.e., voice, text, and clicks) which could explain why Stylette participants reported feeling a higher physical demand (U=127.0, p=0.0187, NP).

6.2 Task 2: Open-Ended Task

To answer RQ2, we evaluated participants' productivity in Task 2 (i.e., how many changes were made and whether varied properties were used). As in Task 1, we also analyzed perceived workload. Samples of the participants' final designs (Fig. 7) show their creativity and how they each focused on different aspects of the website.

6.2.1 Productivity. While participants in the Stylette condition (M=42.85, SD=12.18) made more property changes than those in the DevTools condition (M=39.30, SD=12.71), this difference was not statistically significant (t=0.901, p=0.3729, P). The lack of a statistical difference could be attributed to the benefits and drawbacks of each tool's interaction method. DevTools participants spent more time searching for information, but, once they had the required knowledge, they could directly make changes. Stylette participants could use natural language to easily find properties, but, even if they already knew which property to change, they expended time waiting for the system to process requests and fixing any AI-related

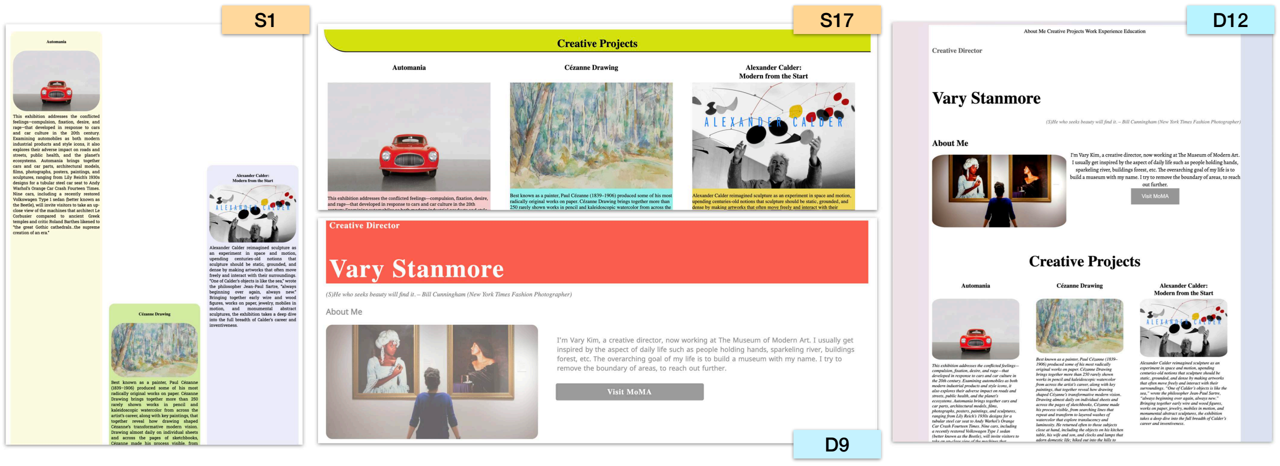


Figure 7: Sample of designs created by Task 2 participants. S1 used *padding* to spread content vertically such that each item would appear gradually as the user scrolls down the page. S17 serendipitously found the *border-width* property and used it to add a "shadow" to the container for the "Creative Projects" subheader. D9 used *opacity* in several components to lighten the web page's content. D12 increased the *border-width* and added *border-color* to add colored bars on the sides of the page.

errors. Several participants (S5, S6, S7, S15) noted that, after learning the properties in Task 1, they wanted to directly change the properties in Task 2—without using natural language.

Additionally, as Stylette presents other options in the *palette*, participants appeared to spend additional time browsing through them. While this exploration could increase effort, it also appeared to encourage familiarization with a wider range of properties. The Gini index for property usage shows that Stylette participants tried various properties (M=0.292, SD=0.045) while DevTools participants mostly stuck with a few properties that they were accustomed to (M=0.325, SD=0.052, t=-2.169, p=0.0364, P). Beyond encouraging experimentation with more properties, in some cases, the system also led participants to serendipitously find alternative uses for known properties. S17 mentioned, *"Accidentally I just found [border-width] while trying to change the radius [so I changed it] and it shows a shadow effect that looks really, really good."* (design shown in Fig. 7).

6.2.2 Perceived Workload. Similar to the results of Task 1, participants in the Stylette condition reported experiencing higher physical demand (U=131.5, p=0.0278, NP) and frustration (U=129.0, p=0.0258, NP) than those in the DevTools condition (Table 4). Unlike Task 1, however, Stylette participants no longer reported feeling significantly less temporal demand or effort. It is plausible that, due to the open-ended nature of Task 2, Stylette participants now spent more time and effort exploring the design space through the alternatives presented by the system—Gini index results support this explanation.

6.2.3 Usage Patterns of Stylette. As Task 2 allowed for more flexible and natural use, we also analyzed participants' usage of Stylette during this task. Participants issued an average of 36.8 requests (max=58, min=18, SD=11.2) and the requests had an average length of 3.2 words (max=12, min=1, SD=1.2).

Our categorization of these requests showed that, unlike our formative study results, the requests were frequently specific and became more specific and less vague towards the end of the task (Table 5). Participants' interviews revealed that this gradual specificity was due to various reasons. For one, the tool helped participants learn property names so they could now use them in requests (S5, S8, S13, S19). Others observed that the system was more accurate if they were more specific, so they adjusted their requests accordingly (S3, S4, S16, S20). A sample of participants' requests (Table 6) shows that the system was indeed more likely to predict users' expected properties if the requests included more specific information.

Like our formative study, however, around half of the requests were vague ("PP", "PV" and "A" in Table 5). Several vague requests were due to participants not remembering the name of a property, but they were able to quickly remember them by seeing Stylette's predicted properties. In other cases, vagueness was to deliberately get the system to act in a certain way. Several participants (S5, S6, S7, S14, S18, S19) mentioned using requests as "macros"—being vague (e.g., "change font") so the system returned several related properties that could be changed in one go. Others (S1, S2, S4, S8, S15) used vague requests to explore what other styling changes they could make.

Regarding the value suggestions, there were three particular uses: (1) as a "starting point", (2) as a "guideline", or (3) as a "shortcut". For the first type, participants (S4, S11, S14, S17, S20) picked a suggested value and then manually adjusted it more to their preference. Others (S2, S5, S9, S10, S18) used the suggestions as a guideline—hovering through values to mentally map numerical differences to visual differences. Finally, as similar values would be suggested for similar components, several participants (S1, S7, S15) looked for the same suggestion when editing multiple similar components—as a sort of "value shortcut".

Type of Request	Description	Examples	Percentage	Q1	Q4
Property Specific (PS)	Specific property name expressed in request.	*"change background color"* *"align text in the center"*	48.1% (352)	46.6%	56.0%
Property Partial (PP)	Property name partially expressed in the request.	*"add border"* *"change the font"*	35.3% (258)	35.2%	34.7%
Property Vague (PV)	Property name not clearly apparent in the request.	*"make this bigger"* *"increase the spacing"*	11.5% (84)	11.9%	4.7%
Property Total	-	-	94.9% (694)	93.8%	95.3%
Value Specific (VS)	Specific value expressed in the request.	*"change to dark grey color"* *"increase font size to 14 px"*	11.4% (83)	14.0%	9.8%
Value Vague (VV)	Vague direction given for a value in the request.	*"decrease the height"* *"make the edges rounder"*	20.0% (146)	25.9%	15.0%
Value Total	-	-	31.2% (229)	39.9%	24.9%
Abstract (A)	Request with abstract description of a change.	*"make it look more stylish"* *"make it more playful"*	3.3% (24)	3.6%	3.1%

Table 5: Coding of the participants' requests during Task 2. Requests can either mention both properties and values, only properties or only values, or be abstract. The percentage of requests for each category are shown. The table also shows the percentage for each category for the first quartile (Q1) and last quartile (Q4) of participants' requests.

6.3 Self-Confidence Across Tasks

To answer RQ3, we evaluated how self-confidence changed during the study by analyzing intra-condition differences in participants' responses (Fig. 8). Stylette participants' self-confidence increased significantly between the pre-survey (M=4.30, SD=1.12) and the end of Task 1 (M=5.13, SD=1.22, z=18.5, p=0.0035, NP). Participants felt satisfied about completing Task 1, and mentioned that it was easy to learn about and make changes using Stylette: *"It gave me the feeling of learning and becoming familiarized with web development terms."* (S12). Surprisingly, DevTools participants' self-confidence also increased significantly between the pre-survey (M=4.01, SD=1.26) and Task 1 (M=4.81, SD=1.25, z=34.5, p=0.0148, NP). Despite most of these participants not completing Task 1, they were satisfied with what they had accomplished as they expected that CSS code would be exceptionally challenging.

For similar reasons, DevTools participants' self-confidence increased between Task 1 (M=4.81, SD=1.25) and Task 2 (M=4.99, SD=1.32), although this was not statistically significant (t=0.540, p=0.595, P). These participants felt proud about their own effort and learning during the study: *"This is my first time handling [CSS] but I did this!"* (D14). In contrast, self-confidence for Stylette participants decreased significantly between Task 1 (M=5.13, SD=1.22) and Task 2 (M=4.58, SD=1.16, t=-3.204, p=0.0047, P). Unlike DevTools participants' self-reflective comments, Stylette participants' comments mostly focused on the tool. Some participants (S9, S16, S17, S20) mentioned how the system presented too many possibilities, making it difficult to decide on changes: *"It was hard [to choose] because the suggestions were all cute."* (S16). On the other hand, several participants (S4, S7, S11, S15) felt limited by the tool's possibilities—expecting the system to reveal new properties or support more complex changes (e.g., adding a "sparkle" animation).

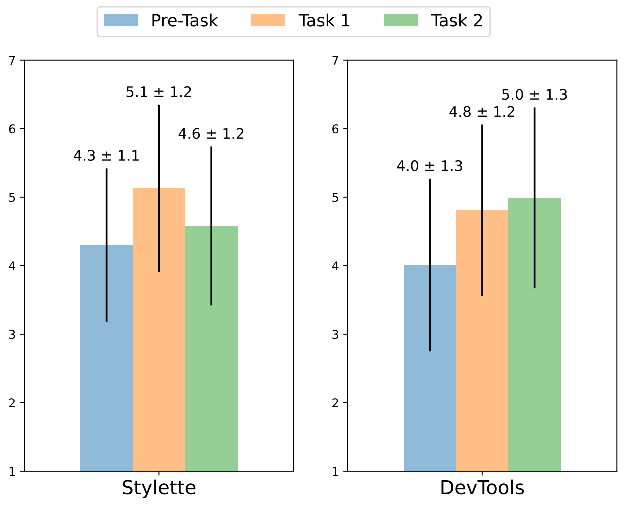


Figure 8: For both conditions, participants' reported self-confidence increased significantly between the pre-survey and the post-Task 1 survey. However, self-confidence decreased significantly for Stylette participants after Task 2, but did not change significantly for DevTools participants.

7 DISCUSSION

In this paper, we propose Stylette, a system that allows users to easily edit a website's design through a suggested set of properties and values generated from natural language requests. Stylette can be generalized to a variety of applications: expanded with a community feature for users to share website modifications, implemented as an IDE plugin to support web developers' help-seeking, or integrated into tools for user feedback. In this section, we further elaborate on the potential of Stylette and suggest opportunities for future work.

7.1 Stylette as a Web Designing Springboard

In our study, Stylette allowed users with no prior knowledge to quickly perform desired styling changes on websites. Unlike search engines that can take the meaning of queries "literally", our system interpreted the vagueness behind users' requests to present more varied and suitable solutions. Stylette also allowed users to "learn-by-doing" by immediately testing the functions of properties by hovering on value suggestions—instead of having to skim through search results. As a side effect of interpreting vagueness, the system appeared to encourage creativity by presenting users with alternatives beyond their initial intentions. Together, these

Request	Type	Expected	Predicted				
"change the font family to Helvetica" (S7)	PS & VS	FF	**FF**	FSz	FSt	FW	
"increase padding" (S8)	PS & VV	P	BW	M	**P**	W	
"change text color" (S6)	PS	C	BgC	**C**	FSt	O	TD
"change the picture radius to 24" (S18)	PP & VS	BR	BC	**BR**	C	FSz	W
"increase the size" (S16)	PP & VV	FSz	BR	BW	H	P	W
"change borders" (S11)	PP	BW	BC	BR	**BW**	C	W
"make it go in the middle" (S15)	PV & VS	TA	H	M	P	W	
"add some spacing at the bottom" (S2)	PV & VV	P	BR	H	M	**P**	
"change the distance" (S5)	PV	M	BW	C	H	P	W
"make this modern" (S19)	A	FF	BW	H	M	P	W

Table 6: A sample of participants' requests in Task 2, ordered from most specific to most vague/abstract. For each property, the table shows the request type, the property expected by the user, and the properties predicted by the system.

insights suggest that Stylette can support novices to explore and learn about CSS with continued usage.

The back-to-back tasks in our study provided a window into such continued usage of Stylette. We observed that users gradually developed knowledge about concrete CSS property names and values. For these now more knowledgeable users, the system still provided benefit: enabling the request of multiple properties for increased efficiency, supporting exploration of the design space, and helping users quickly remember forgotten information. However, we also observed that perceived effort could increase with continued usage and users' learning. This owed to the fact that, even after acquiring the knowledge to directly make changes by themselves, user still had to interact with the underlying, probabilistic AI—waiting for its processing and correcting any errors.

Thus, while Stylette is well-suited for novices to learn about CSS, its benefit may decrease with users' increasing knowledge due to the form of interaction. Elaborating on Amershi et al.'s guidelines [3], this suggests how human-AI interaction should be designed for over time use in the context of novice support systems. For future work, we propose an adaptive approach: initial natural language interaction to help users acquire knowledge about properties, and then gradually exposing direct manipulation widgets for properties that users have acquired knowledge about. Knowledge could be modeled by identifying previously used properties, repeated usage of a property, or the use of the property's name in voice requests.

To further overcome the frustration and physical demand observed in the study, future work could also investigate mechanisms to support the discoverability of natural language input. Prior work [14, 23, 56] has demonstrated that supporting discoverability can reduce the amount of "guessing" that users must do. As Stylette's NLP pipeline appears to provide more accurate predictions for specific requests, future iterations of the system could guide users to new or desired properties by suggesting more specific language. For example, if the user makes a vague request but does not use any of the predicted properties, the system could suggest specific requests related to other properties that the user has not seen before.

7.2 Leveraging Large Language Models to Support Software Use

The grand scale of large language models (e.g., GPT-3 [8] or GPT-Neo [6]), in terms of architecture and datasets, has allowed them to perform previously unseen tasks with only a few data points. We leveraged this quality and the P-tuning technique [42] to allow novices to interact with website designs by constructing only a small dataset of 300 requests. Similar approaches can be taken to enable novices to use natural language to use various complex software—overcoming the vocabulary problem [22]. While a rich body of work has enabled similar natural language interaction to support software usage [1, 18–21], their approaches relied on a wealth of user-generated content. Thus, these approaches are not possible for new applications or features as such content might not exist. Moreover, as shown by the struggles of DevTools participants in our study, the language used in such content may also differ greatly from the vague language used by novices as the content is usually created by intermediate or advanced users. With our approach, in contrast, natural language interaction can be enabled for new applications with only the effort of creating a small dataset of examples, and, by including representative examples of novices' language, the support can be designed specifically for novices.

7.3 Natural Language Coding as a Learning Tool

Our natural language interface helps novices learn about a coding language by demonstrating how the code realizes high-level goals—lowering the selection, coordination, and use barriers identified by Ko et al. [31]. In addition, by exposing novices to multiple alternatives for an intended goal, we observed that our approach allowed users to acquire a greater breadth of knowledge about the code—familiarizing with more properties and learning new uses for properties. However, the study also revealed that DevTools participants appeared to feel more satisfaction about their learning experience when compared to Stylette. We suspect that this is due to DevTools participants expending more deliberate effort searching for and reading through resources. Based on these insights, we first suggest that natural language coding tools should provide multiple code alternatives for the same goal. Then, by incorporating interventions that prompt users to reflect on these alternatives—similar to prompts used in video learning [55]—to gain a wider understanding about the code through a deliberate learning experience.

7.4 Beyond CSS

Stylette aims to make the web more malleable for general users with no prior knowledge. Our work focuses on CSS code and allows novices to simply describe their high-level goal to start modifying it—without requiring the user to decompose the goal themselves [58] or look for examples [16, 33, 37]. However, websites are also composed of HTML (structure) and JavaScript code (functionalities). As structure-related changes might be more suitable for direct manipulation, Stylette could be combined with systems that already support this [46, 47]. Finally, to allow end-users to program new functionalities, models like OpenAI's Codex [62], which can generate JavaScript code from natural language descriptions, could be coupled with Stylette. By integrating these three types of support into one coherent system, future work could enable all users to fully access the web's malleability.

8 LIMITATIONS

Our work has several limitations which we address in this section.

- Stylette currently supports 16 different CSS properties. These were the ones used the most in the creation of our request dataset. While Stylette could be extended to support more properties by expanding the dataset, certain complex properties (e.g., those related to flexbox and grid) also require corresponding modifications on parent elements. As Stylette only modifies the selected element's properties, it cannot currently support these properties. To overcome this limitation, the system could be enhanced to cascade necessary property modifications up the HTML tree.
- In our evaluation, we compared Stylette against using DevTools and search engines. A possible concern is that DevTools participants could change more properties and might have

misdirected effort into these. Although the average DevTools participant only tried around two properties that were not supported in Stylette, we acknowledge that this could have affected results in Task 1.

- We relied on a dataset of 300 requests to train and evaluate our computational pipeline. While participants were generally satisfied with the pipeline's predictions, evaluating on a larger dataset would provide a better understanding of its performance. Also, while P-tuning has demonstrated high performance with even smaller datasets (N=32) [42], a larger dataset could increase our pipeline's performance and robustness.
- As we focused on a controlled evaluation of Stylette, it is still unclear how users would modify websites in the real-world. Future work could conduct a deployment study to understand how Stylette integrates into users' actual web experiences.

9 CONCLUSION

This paper presents *Stylette*, a novel system that allows users to describe a styling request in natural language to change the visual design of websites. By combining a GPT-Neo-based model and a convolutional VAE model, our computational pipeline processes the user's request and the component they clicked. The processed outputs are then combined to generate a *palette* of CSS properties and values that the user can experiment and iterate on to reach their desired style. A user-study revealed that Stylette could help users familiarize themselves with CSS properties in a shorter amount of time and with greater breadth. Insights from the study regarding the benefits and limitations of natural language support can guide the design of future work on novice support systems.

ACKNOWLEDGMENTS

This work was supported by Institute of Information & communications Technology Planning & Evaluation(IITP) grant funded by the Korea government (MSIT) (No.2021-0-01347,Video Interaction Technologies Using Object-Oriented Video Modeling). This work was partly supported by KAIST-NAVER Hypercreative AI Center. The authors would like to thank the members of KIXLAB for their thoughtful comments and thank our participants for their positive engagement during the studies. Finally, we thank the reviewers as their feedback helped us improve the paper.

REFERENCES

[1] Eytan Adar, Mira Dontcheva, and Gierad Laput. 2014. CommandSpace: Modeling the Relationships between Tasks, Descriptions and Features. In *Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology* (Honolulu, Hawaii, USA) *(UIST '14)*. Association for Computing Machinery, New York, NY, USA, 167–176. https://doi.org/10.1145/2642918.2647395

[2] Miltos Allamanis, Daniel Tarlow, Andrew Gordon, and Yi Wei. 2015. Bimodal Modelling of Source Code and Natural Language. In *Proceedings of the 32nd International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 37)*, Francis Bach and David Blei (Eds.). PMLR, Lille, France, 2123–2132. https://proceedings.mlr.press/v37/allamanis15.html

[3] Saleema Amershi, Dan Weld, Mihaela Vorvoreanu, Adam Fourney, Besmira Nushi, Penny Collisson, Jina Suh, Shamsi Iqbal, Paul N. Bennett, Kori Inkpen, Jaime Teevan, Ruth Kikin-Gil, and Eric Horvitz. 2019. Guidelines for Human-AI Interaction. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. Association for Computing Machinery, New York, NY, USA, 1–13. https://doi.org/10.1145/3290605.3300233

[4] The Webby Awards. 2021. Top Websites and Mobile Sites | The Webby Awards. Retrieved August 29, 2021 from https://winners.webbyawards.com/winners/websites-and-mobile-sites

[5] Jan Biniok. 2021. Tampermonkey. Retrieved September 5, 2021 from https://www.tampermonkey.net/

[6] Sid Black, Leo Gao, Phil Wang, Connor Leahy, and Stella Biderman. 2021. *GPT-Neo: Large Scale Autoregressive Language Modeling with Mesh-Tensorflow.* https://doi.org/10.5281/zenodo.5297715 If you use this software, please cite it using these metadata.

[7] Michael Bolin, Matthew Webber, Philip Rha, Tom Wilson, and Robert C. Miller. 2005. Automation and Customization of Rendered Web Pages. In *Proceedings of the 18th Annual ACM Symposium on User Interface Software and Technology* (Seattle, WA, USA) *(UIST '05)*. Association for Computing Machinery, New York, NY, USA, 163–172. https://doi.org/10.1145/1095034.1095062

[8] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language Models are Few-Shot Learners. arXiv:2005.14165 [cs.CL]

[9] Kerry Shih-Ping Chang and Brad A. Myers. 2012. *WebCrystal: Understanding and Reusing Examples in Web Authoring*. Association for Computing Machinery, New York, NY, USA, 3205–3214. https://doi.org/10.1145/2207676.2208740

[10] Siddhartha Chaudhuri, Evangelos Kalogerakis, Stephen Giguere, and Thomas Funkhouser. 2013. Attribit: Content Creation with Semantic Attributes. In *Proceedings of the 26th Annual ACM Symposium on User Interface Software and Technology* (St. Andrews, Scotland, United Kingdom) *(UIST '13)*. Association for Computing Machinery, New York, NY, USA, 193–202. https://doi.org/10.1145/2501988.2502008

[11] Chunyang Chen, Sidong Feng, Zhenchang Xing, Linda Liu, Shengdong Zhao, and Jinshui Wang. 2019. Gallery D.C.: Design Search and Knowledge Discovery through Auto-Created GUI Component Gallery. *Proc. ACM Hum.-Comput. Interact.* 3, CSCW, Article 180 (Nov. 2019), 22 pages. https://doi.org/10.1145/3359282

[12] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. 2021. Evaluating Large Language Models Trained on Code. arXiv:2107.03374 [cs.LG]

[13] Yan Chen, Sang Won Lee, and Steve Oney. 2021. CoCapture: Effectively Communicating UI Behaviors on Existing Websites by Demonstrating and Remixing. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems* (Yokohama, Japan) *(CHI '21)*. Association for Computing Machinery, New York, NY, USA, Article 416, 14 pages. https://doi.org/10.1145/3411764.3445573

[14] Eric Corbett and Astrid Weber. 2016. What Can I Say? Addressing User Experience Challenges of a Mobile Voice User Interface for Accessibility. In *Proceedings of the 18th International Conference on Human-Computer Interaction with Mobile Devices and Services* (Florence, Italy) *(MobileHCI '16)*. Association for Computing Machinery, New York, NY, USA, 72–82. https://doi.org/10.1145/2935334.2935386

[15] DomCop. 2021. What is Open PageRank? Retrieved August 29, 2021 from https://www.domcop.com/openpagerank/what-is-openpagerank

[16] Michael J Fitzgerald et al. 2008. *CopyStyler: Web design by example.* Ph.D. Dissertation. Massachusetts Institute of Technology.

[17] James Fogarty, Desney Tan, Ashish Kapoor, and Simon Winder. 2008. CueFlik: Interactive Concept Learning in Image Search. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Florence, Italy) *(CHI '08)*. Association for Computing Machinery, New York, NY, USA, 29–38. https://doi.org/10.1145/1357054.1357061

[18] Adam Fourney, Richard Mann, and Michael Terry. 2011. Query-Feature Graphs: Bridging User Vocabulary and System Functionality. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology* (Santa Barbara, California, USA) *(UIST '11)*. Association for Computing Machinery, New York, NY, USA, 207–216. https://doi.org/10.1145/2047196.2047224

[19] C. Ailie Fraser, Mira Dontcheva, Holger Winnemöller, Sheryl Ehrlich, and Scott Klemmer. 2016. DiscoverySpace: Suggesting Actions in Complex Software. In *Proceedings of the 2016 ACM Conference on Designing Interactive Systems* (Brisbane, QLD, Australia) *(DIS '16)*. Association for Computing Machinery, New York, NY, USA, 1221–1232. https://doi.org/10.1145/2901790.2901849

[20] C. Ailie Fraser, Julia M. Markel, N. James Basa, Mira Dontcheva, and Scott Klemmer. 2020. ReMap: Lowering the Barrier to Help-Seeking with Multimodal Search.

In *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology* (Virtual Event, USA) *(UIST '20)*. Association for Computing Machinery, New York, NY, USA, 979–986. https://doi.org/10.1145/3379337.3415592

[21] C. Ailie Fraser, Tricia J. Ngoon, Mira Dontcheva, and Scott Klemmer. 2019. RePlay: Contextually Presenting Learning Videos Across Software Applications. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. Association for Computing Machinery, New York, NY, USA, 1–13. https://doi.org/10.1145/3290605.3300527

[22] G. W. Furnas, T. K. Landauer, L. M. Gomez, and S. T. Dumais. 1987. The Vocabulary Problem in Human-System Communication. *Commun. ACM* 30, 11 (Nov. 1987), 964–971. https://doi.org/10.1145/32206.32212

[23] Anushay Furqan, Chelsea Myers, and Jichen Zhu. 2017. Learnability through Adaptive Discovery Tools in Voice User Interfaces. In *Proceedings of the 2017 CHI Conference Extended Abstracts on Human Factors in Computing Systems* (Denver, Colorado, USA) *(CHI EA '17)*. Association for Computing Machinery, New York, NY, USA, 1617–1623. https://doi.org/10.1145/3027063.3053166

[24] Tong Gao, Mira Dontcheva, Eytan Adar, Zhicheng Liu, and Karrie G. Karahalios. 2015. DataTone: Managing Ambiguity in Natural Language Interfaces for Data Visualization. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology* (Charlotte, NC, USA) *(UIST '15)*. Association for Computing Machinery, New York, NY, USA, 489–500. https://doi.org/10.1145/2807442.2807478

[25] Greasemonkey. 2021. Greasespot. Retrieved September 5, 2021 from https://www.greasespot.net/

[26] Sandra G Hart and Lowell E Staveland. 1988. Development of NASA-TLX (Task Load Index): Results of empirical and theoretical research. In *Advances in psychology*, Vol. 52. Elsevier, 139–183.

[27] Dan Hendrycks, Steven Basart, Saurav Kadavath, Mantas Mazeika, Akul Arora, Ethan Guo, Collin Burns, Samir Puranik, Horace He, Dawn Song, and Jacob Steinhardt. 2021. Measuring Coding Challenge Competence With APPS. arXiv:2105.09938 [cs.SE]

[28] Jane Im, Sonali Tandon, Eshwar Chandrasekharan, Taylor Denby, and Eric Gilbert. 2020. *Synthesized Social Signals: Computationally-Derived Social Signals from Account Histories*. Association for Computing Machinery, New York, NY, USA, 1–12. https://doi.org/10.1145/3313831.3376383

[29] Youwen Kang, Zhida Sun, Sitong Wang, Zeyu Huang, Ziming Wu, and Xiaojuan Ma. 2021. MetaMap: Supporting Visual Metaphor Ideation through Multi-Dimensional Example-Based Exploration. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems* (Yokohama, Japan) *(CHI '21)*. Association for Computing Machinery, New York, NY, USA, Article 427, 15 pages. https://doi.org/10.1145/3411764.3445325

[30] Diederik P Kingma and Max Welling. 2014. Auto-Encoding Variational Bayes. arXiv:1312.6114 [stat.ML]

[31] Amy J. Ko, Brad A. Myers, and Htet Htet Aung. 2004. Six Learning Barriers in End-User Programming Systems. In *2004 IEEE Symposium on Visual Languages - Human Centric Computing*. IEEE, 199–206. https://doi.org/10.1109/VLHCC.2004.47

[32] Ranjitha Kumar, Arvind Satyanarayan, Cesar Torres, Maxine Lim, Salman Ahmad, Scott R. Klemmer, and Jerry O. Talton. 2013. Webzeitgeist: Design Mining the Web. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. Association for Computing Machinery, New York, NY, USA, 3083–3092. https://doi.org/10.1145/2470654.2466420

[33] Ranjitha Kumar, Jerry O. Talton, Salman Ahmad, and Scott R. Klemmer. 2011. *Bricolage: Example-Based Retargeting for Web Design*. Association for Computing Machinery, New York, NY, USA, 2197–2206. https://doi.org/10.1145/1978942.1979262

[34] Google Chrome Labs. 2018. VisBug. Retrieved August 26, 2021 from https://visbug.web.app/

[35] Michelle S. Lam, Grace B. Young, Catherine Y. Xu, Ranjay Krishna, and Michael S. Bernstein. 2019. Eevee: Transforming Images by Bridging High-Level Goals and Low-Level Edit Operations. In *Extended Abstracts of the 2019 CHI Conference on Human Factors in Computing Systems* (Glasgow, Scotland Uk) *(CHI EA '19)*. Association for Computing Machinery, New York, NY, USA, 1–6. https://doi.org/10.1145/3290607.3312929

[36] Gierad P. Laput, Mira Dontcheva, Gregg Wilensky, Walter Chang, Aseem Agarwala, Jason Linder, and Eytan Adar. 2013. *PixelTone: A Multimodal Interface for Image Editing*. Association for Computing Machinery, New York, NY, USA, 2185–2194. https://doi.org/10.1145/2470654.2481301

[37] Brian Lee, Savil Srivastava, Ranjitha Kumar, Ronen Brafman, and Scott R. Klemmer. 2010. *Designing with Interactive Example Galleries*. Association for Computing Machinery, New York, NY, USA, 2257–2266. https://doi.org/10.1145/1753326.1753667

[38] Toby Jia-Jun Li, Marissa Radensky, Justin Jia, Kirielle Singarajah, Tom M. Mitchell, and Brad A. Myers. 2019. PUMICE: A Multi-Modal Agent That Learns Concepts and Conditionals from Natural Language and Demonstrations. In *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology* (New Orleans, LA, USA) *(UIST '19)*. Association for Computing Machinery, New York, NY, USA, 577–589. https://doi.org/10.1145/3332165.3347899

[39] Sarah Lim, Joshua Hibschman, Haoqi Zhang, and Eleanor O'Rourke. 2018. Ply: A Visual Web Inspector for Learning from Professional Webpages. In *Proceedings of the 31st Annual ACM Symposium on User Interface Software and Technology* (Berlin, Germany) *(UIST '18)*. Association for Computing Machinery, New York, NY, USA, 991–1002. https://doi.org/10.1145/3242587.3242660

[40] Xi Victoria Lin, Chenglong Wang, Luke Zettlemoyer, and Michael D. Ernst. 2018. NL2Bash: A Corpus and Semantic Parser for Natural Language Interface to the Linux Operating System. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*. European Language Resources Association (ELRA), Miyazaki, Japan. https://aclanthology.org/L18-1491

[41] Wang Ling, Phil Blunsom, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiský, Fumin Wang, and Andrew Senior. 2016. Latent Predictor Networks for Code Generation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Berlin, Germany, 599–609. https://doi.org/10.18653/v1/P16-1057

[42] Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. 2021. GPT Understands, Too. arXiv:2103.10385 [cs.CL]

[43] Edward Ma. 2019. NLP Augmentation. https://github.com/makcedward/nlpaug.

[44] Mehdi Manshadi, Daniel Gildea, and James Allen. 2013. Integrating Programming by Example and Natural Language Programming. https://www.aaai.org/ocs/index.php/AAAI/AAAI13/paper/view/6477/7230

[45] Rada Mihalcea, Hugo Liu, and Henry Lieberman. 2006. NLP (Natural Language Processing) for NLP (Natural Language Programming). In *Computational Linguistics and Intelligent Text Processing*, Alexander Gelbukh (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 319–330.

[46] Michael Nebeling and Anind K. Dey. 2016. *XDBrowser: User-Defined Cross-Device Web Page Designs*. Association for Computing Machinery, New York, NY, USA, 5494–5505. https://doi.org/10.1145/2858036.2858048

[47] Michael Nebeling, Maximilian Speicher, and Moira C. Norrie. 2013. CrowdAdapt: Enabling Crowdsourced Web Page Adaptation for Individual Viewing Conditions and Preferences. In *Proceedings of the 5th ACM SIGCHI Symposium on Engineering Interactive Computing Systems* (London, United Kingdom) *(EICS '13)*. Association for Computing Machinery, New York, NY, USA, 23–32. https://doi.org/10.1145/2494603.2480304

[48] Jonas Oppenlaender, Thanassis Tiropanis, and Simo Hosio. 2020. CrowdUI: Supporting Web Design with the Crowd. *Proc. ACM Hum.-Comput. Interact.* 4, EICS, Article 76 (June 2020), 28 pages. https://doi.org/10.1145/3394978

[49] Thomas H. Park, Ankur Saxena, Swathi Jagannath, Susan Wiedenbeck, and Andrea Forte. 2013. OpenHTML: Designing a Transitional Web Editor for Novices. In *CHI '13 Extended Abstracts on Human Factors in Computing Systems* (Paris, France) *(CHI EA '13)*. Association for Computing Machinery, New York, NY, USA, 1863–1868. https://doi.org/10.1145/2468356.2468690

[50] Chris Quirk, Raymond Mooney, and Michel Galley. 2015. Language to Code: Learning Semantic Parsers for If-This-Then-That Recipes. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Association for Computational Linguistics, Beijing, China, 878–888. https://doi.org/10.3115/v1/P15-1085

[51] Daniel Ritchie, Ankita Arvind Kejriwal, and Scott R. Klemmer. 2011. D.Tour: Style-Based Exploration of Design Example Galleries. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology* (Santa Barbara, California, USA) *(UIST '11)*. Association for Computing Machinery, New York, NY, USA, 165–174. https://doi.org/10.1145/2047196.2047216

[52] Xin Rong, Shiyan Yan, Stephen Oney, Mira Dontcheva, and Eytan Adar. 2016. CodeMend: Assisting Interactive Programming with Bimodal Embedding. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology* (Tokyo, Japan) *(UIST '16)*. Association for Computing Machinery, New York, NY, USA, 247–258. https://doi.org/10.1145/2984511.2984544

[53] Viktor Schlegel, Benedikt Lang, Siegfried Handschuh, and André Freitas. 2019. Vajra: Step-by-Step Programming with Natural Language. In *Proceedings of the 24th International Conference on Intelligent User Interfaces* (Marina del Ray, California) *(IUI '19)*. Association for Computing Machinery, New York, NY, USA, 30–39. https://doi.org/10.1145/3301275.3302267

[54] Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Improving Neural Machine Translation Models with Monolingual Data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Berlin, Germany, 86–96. https://doi.org/10.18653/v1/P16-1009

[55] Hyungyu Shin, Eun-Young Ko, Joseph Jay Williams, and Juho Kim. 2018. *Understanding the Effect of In-Video Prompting on Learners and Instructors*. Association for Computing Machinery, New York, NY, USA, 1–12. https://doi.org/10.1145/3173574.3173893

[56] Arjun Srinivasan, Mira Dontcheva, Eytan Adar, and Seth Walker. 2019. Discovering Natural Language Commands in Multimodal Interfaces. In *Proceedings of the 24th International Conference on Intelligent User Interfaces* (Marina del Ray, California) *(IUI '19)*. Association for Computing Machinery, New York, NY, USA, 661–672. https://doi.org/10.1145/3301275.3302292

[57] Amanda Swearngin, Amy J. Ko, and James Fogarty. 2017. *Genie: Input Retargeting on the Web through Command Reverse Engineering.* Association for Computing Machinery, New York, NY, USA, 4703–4714. https://doi.org/10.1145/3025453.3025506

[58] Kesler Tanner, Naomi Johnson, and James A. Landay. 2019. *Poirot: A Web Inspector for Designers.* Association for Computing Machinery, New York, NY, USA, 1–12. https://doi.org/10.1145/3290605.3300758

[59] Sebastian Weigelt, Vanessa Steurer, Tobias Hey, and Walter F. Tichy. 2020. Programming in Natural Language with fuSE: Synthesizing Methods from Spoken Utterances Using Deep Natural Language Understanding. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics.* Association for Computational Linguistics, Online, 4280–4295. https://doi.org/10.18653/v1/2020.acl-main.395

[60] Haijun Xia. 2020. Crosspower: Bridging Graphics and Linguistics. In *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology* (Virtual Event, USA) *(UIST '20).* Association for Computing Machinery, New York, NY, USA, 722–734. https://doi.org/10.1145/3379337.3415845

[61] Pengcheng Yin and Graham Neubig. 2017. A Syntactic Neural Model for General-Purpose Code Generation. In *The 55th Annual Meeting of the Association for Computational Linguistics (ACL).* Vancouver, Canada. https://arxiv.org/abs/1704.01696

[62] Wojciech Zaremba, Greg Brockman, and OpenAI. 2021. OpenAI Codex. Retrieved August 28, 2021 from https://openai.com/blog/openai-codex/

[63] Xiong Zhang and Philip J. Guo. 2018. Fusion: Opportunistic Web Prototyping with UI Mashups. In *Proceedings of the 31st Annual ACM Symposium on User Interface Software and Technology* (Berlin, Germany) *(UIST '18).* Association for Computing Machinery, New York, NY, USA, 951–962. https://doi.org/10.1145/3242587.3242632

[64] Xiang Zhang, Junbo Jake Zhao, and Yann LeCun. 2015. Character-level Convolutional Networks for Text Classification. In *NIPS.* 649–657. http://papers.nips.cc/paper/5782-character-level-convolutional-networks-for-text-classification

[65] Nanxuan Zhao, Nam Wook Kim, Laura Mariah Herman, Hanspeter Pfister, Rynson W.H. Lau, Jose Echevarria, and Zoya Bylinskii. 2020. *ICONATE: Automatic Compound Icon Generation and Ideation.* Association for Computing Machinery, New York, NY, USA, 1–13. https://doi.org/10.1145/3313831.3376618

[66] Tony Z. Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. 2021. Calibrate Before Use: Improving Few-Shot Performance of Language Models. arXiv:2102.09690 [cs.CL]

[67] Mingyuan Zhong, Gang Li, and Yang Li. 2021. Spacewalker: Rapid UI Design Exploration Using Lightweight Markup Enhancement and Crowd Genetic Programming. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems* (Yokohama, Japan) *(CHI '21).* Association for Computing Machinery, New York, NY, USA, Article 315, 11 pages. https://doi.org/10.1145/3411764.3445326

[68] Victor Zhong, Caiming Xiong, and Richard Socher. 2017. Seq2SQL: Generating Structured Queries from Natural Language using Reinforcement Learning. arXiv:1709.00103 [cs.CL]