A Blocks-Based Editor for HTML Code

Saksham Aggarwal
International Institue of Information Technology
Hyderabad, 500032
Email: saksham.aggarwal@students.iiit.ac.in

David Anthony Bau
Phillips Exeter Academy
Exeter, New Hampshire 03833
Email: dbau@exeter.edu

***Abstract*—Droplet is a new dual-mode editor that allows students to work in either blocks or text and switch between them any time. This paper presents work creating a Droplet mode for HTML code. We also discuss an analysis of real-world HTML tags and attributes and propose a palette based on this analysis.**

I. Introduction

Teaching HTML has long been an early step in a programming curriculum. For example Budny, et al [2] in Four Steps to Teaching C Programming, suggest "The layout of a web page allowed us to begin to teach the basic concepts of program layout... We are teaching web page design ... not for the purpose of teaching HTML, but to teach students the concept of writing code." Mahmoud, et al [3] suggest that starting with HTML is a way of teaching "programming for fun" and is a strategy for motivating students.

Nonetheless, for first-time-coder, HTML can be difficult to learn. In a workshop with English students, Mauriello, Pagnucci, and Winner [4] observed "Students are generally not careful and experienced enough in their reading of the codes to find mistakes." For non-coding students, Taylor and Gitsaki [5] suggest simplifying the problem by starting with a small set of about 30 HTML tags to create a basic web page.

Therefore we are interested in finding an alternative to WYSIWYG HTML tools that expose the code, while still simplifying the process of learning to use HTML tags for the first time. In recent years, block programming languages such as Scratch [6] have introduced many students to coding through a visual representation of commands and control flow. Here we investigate whether a similar approach can be effective when used with HTML code.

II. Background

A. Droplet's Text-First Approach to Blocks

Droplet [1] is a dual-mode blocks and text editor that was built to bridge the gap between blocks and text. Droplet's primary guiding philosophy is that the text, not the blocks, are the primary data. Thus, Droplet programs begin and end their life as text. When Droplet opens a program file, the language adapter inserts markup indicating where blocks should go and how they should be rendered. The user interacts with this rendering of the program, performing splice operations on the markup stream. During editing, the language mode may be called back to preserve precedence or dictate droppability rules. At the end of the editing session, the markup is simply discarded and a raw text program is generated again. Figure 1 shows a typical Droplet editing session in JavaScript.

B. Adding A New Language to Droplet

A Droplet language adapter has two roles: to parse text and insert block markup, and to enforce droppability rules between blocks and sockets. Usually, a Droplet language adapter uses a standard language parser – for instance, Droplet's JavaScript mode uses acorn.js – and inserts blocks using the location data from the generated AST. The parser annotates the generated blocks with information pertinent to droppability, and uses this information later during editing to determine whether a drop is legal.

III. Process

A. Adapting A Parser

One of the goals of an HTML mode in Droplet is to be able to visualize existing webpages from the Internet as blocks. This poses a difficulty because browsers are tolerant and many existing webpages are not standards-compliant or are syntactically incorrect. Droplet's HTML mode adapts the parse5 [7] HTML parser, which tolerates syntactically incorrect HTML code in the same way browsers do. The parse5 parser was modified for Droplet's purposes to add more detailed location data. Here is how it works:

1) Parse the text into an Abstract Syntax Tree using parse5.
2) Mark the root of the document.
3) For each node, check if it is a text node, comment, empty tag or a compound tag.
4) Mark the node based on its type
   - If it is a text node, make it editable.
   - If it is a comment node, make the comment editable.
   - If it is an empty tag, mark it as a block and make its attributes editable.
   - If is a compound tag, mark it as a block, make its attributes editable and add an indent for its children.
5) Is the node has children, recurse from step (3) for every child.

B. Enforcing Droppability Rules

One major advantage of a block language, however, is that it can enforce creating only standards-compliant code. Droplet's HTML mode therefore enforces droppability rules adapted from the WHATWG HTML specifications [8]. Here are the implemented rules for some tags about what is allowed immediately inside the tag

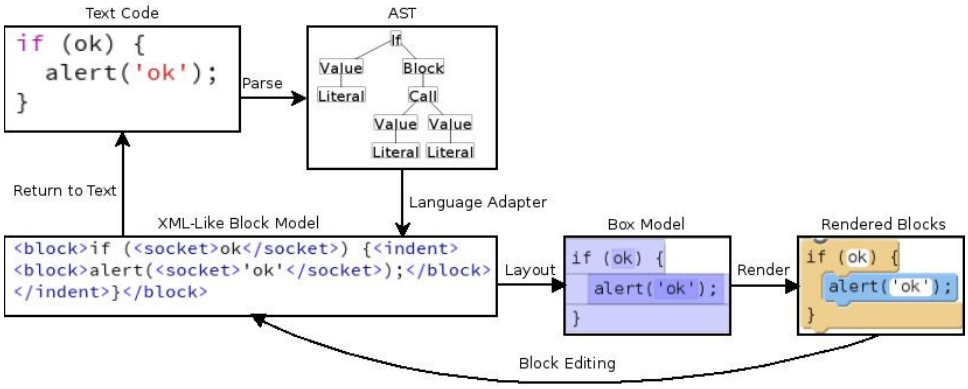

Fig. 1. Lifecycle of a Droplet Program

html - [head? (body|frameset)?]
head - [METADATA_CONTENT*] [19]
title - ['text'?]
a - [(PHRASING_CONTENT|INTERACTIVE_CONTENT)*] [20] [21]
body - [FLOW_CONTENT*] [18]
table - [(caption|colgroup|thead|tfoot|tbody|tr| SCRIPT_SUPPORTING)*] [22]
td - [FLOW_CONTENT*] [18]
Similarly the rules were implemented for 92 tags adapting from standards set by WHATWG [8]

C. Choosing A Palette

According to Whoever [**?**], the palette in a block language is important to discovery and self-directed learning, because students can try new commands without having to read documentation. Having a palette that contains useful and rewarding tags in an HTML mode is therefore important. The WHATWG HTML specifications define over 100 tags, however, most of which are not used. A number of developers online have informally posted HTML cheat sheets with the "most important tags," [12] [13] [14] but these are subjective and often conflict with each other. Because Droplet's philosophy is to be able to interact with real-world code on the Internet, we here determine and recommend a palette based on real-world tag frequencies.

Figures 2 and 3 show an analysis of tag frequencies over random HTML datasets collected from commoncrawl [10] (data and full results are available on Github [9]. Figure 2 represents a count of the number of times each tag was used at all in the crawled data sets. Figure 3 represents a count of the number of documents in the data sets that used the tag.

The final palette [11] was created by choosing top 40 tags from the above 2 analysis results. Added to this were inputs from teachers and courses they like to start with to teach HTML, and some starter courses which children do themselves [15] [16] [17]. You may notice that the palette isn't exactly the top 40 tags but this is because starters need not learn all highly used tags, they start with tags which are easier to understand.

IV. Future Prospects

A. Palette with alternate blocks

As of now, the palette has some commonly used tags. This can be made better by including variations of tags based on commonly used attributes. An example can be inclusion of both, <script></script>block and <script src='uri' ></script>block. A detailed study was done of the commonly used attributes for every tag as can be found on GitHub [9]. The results lists down commonly used attribtutes for all the tags, again sampling randomly over real world HTML collected from commoncrawl.

References

[1] Bau, D. A. Droplet, A Blocks-Based Editor for Text Code. Journal of Computer Science in Colleges. 30, 6 (June 2015).

[2] Budny, D.; Lund, L.; Vipperman, J.; Patzer, J.L.I.I.I., "Four steps to teaching C programming," Frontiers in Education, 2002. FIE 2002. 32nd Annual , vol.2, no., pp.F1G-18,F1G-22 vol.2, 2002

[3] Qusay H. Mahmoud, Wlodek Dobosiewicz, and David Swayne. 2004. Redesigning introductory computer programming with HTML, JavaScript, and Java. SIGCSE Bull. 36, 1 (March 2004), 120-124. DOI=10.1145/1028174.971344 http://doi.acm.org/10.1145/1028174.971344

[4] Mauriello, N. Pagnucci, G. and Winner, T. Reading between the Code: The Teaching of HTML and the Displacement of Writing Instruction. Computers and Composition 16, 409-19 (1999)

[5] Taylor, R. and Gitaski, C. Teaching WELL and loving IT. New Perspectives on CALL for Second Language Classrooms, 131-147.

[6] Scratch. https://scratch.mit.edu/

[7] Parse5. https://github.com/inikulin/parse5

[8] HTML living standard. https://html.spec.whatwg.org/

[9] https://github.com/sakagg/HTMLtagsFrequencyAnalysis

[10] Common Crawl. https://commoncrawl.org/

[11] HTML Palette in use on Pencil Code. pencilcode.net/edit/example.html

[12] Webmonkey. HTML Cheat Sheet. http://www.webmonkey.com/2010/02/html_cheatsl

[13] A Simple Guide to HTML. HTML Cheat Sheet. http://www.simplehtmlguide.com/cheatsheet.php

[14] Usabilla. An HTML Cheat Sheet That Never Fails. http://blog.usabilla.com/an-html-cheat-sheet-that-never-fails/

[15] Exploring computer Science - pages 105-110 http://www.exploringcs.org/wp-content/uploads/2014/02/ExploringComputerScience-v5.0.pdf

[16] W3Schools HTML starters guide http://www.w3schools.com/html/default.asp

[17] Htmldog beginner tutorial http://htmldog.com/guides/html/beginner/conclusion/

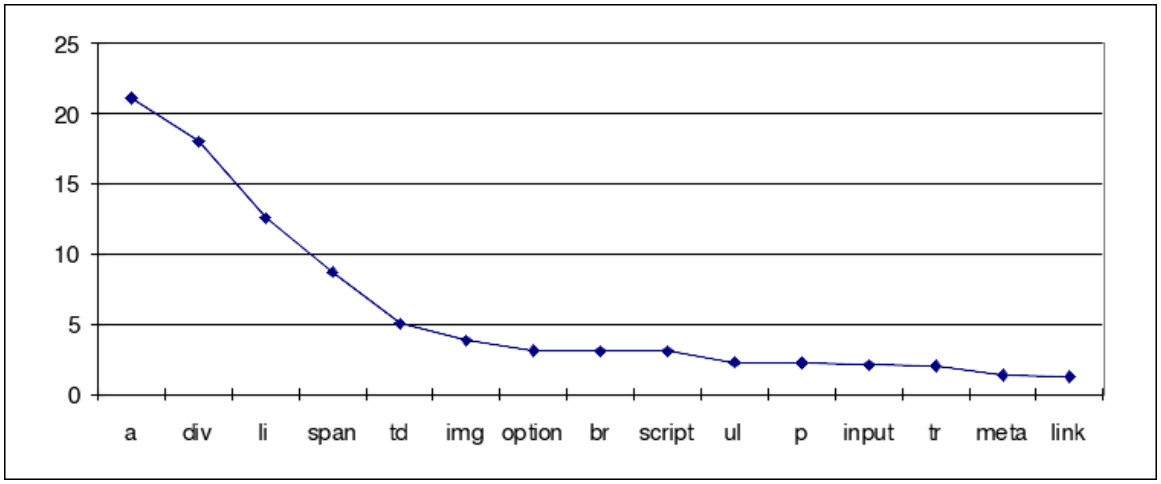

Fig. 2. Counts Based On Tag Appearances

Y-axis - percentage occurance of the tag

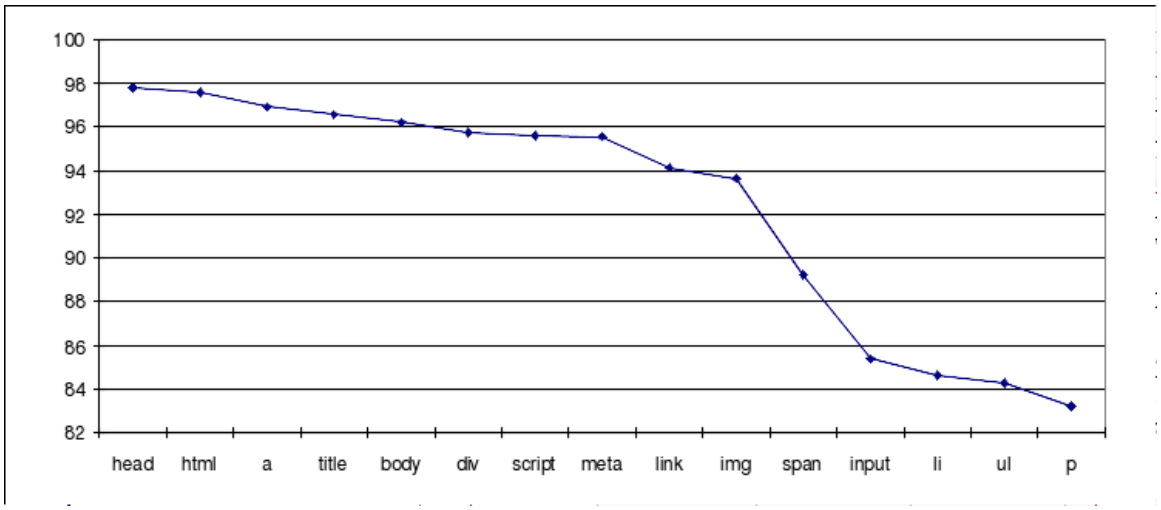

Fig. 3. Counts Based on Documents With Tag

Y-axis - percentage of documents having the tag

[18] WHATWG flow content list https://developers.whatwg.org/content-models.htmlflow-content

[19] WHATWG metadata content list https://developers.whatwg.org/content-models.htmlmetadata-content

[20] WHATWG phrasing content list https://developers.whatwg.org/content-models.htmlphrasing-content

[21] WHATWG interactive content list https://developers.whatwg.org/content-models.htmlinteractive-content

[22] WHATWG script supporting elements https://developers.whatwg.org/content-models.htmlscript-supporting-elements