Lessons from a Web-Based IDE and Runtime

Manuel Fähndrich
Microsoft Research
maf@microsoft.com

Abstract

At Microsoft Research, we have built a purely web-based IDE called TouchDevelop that enables anyone to pick up a device and start programming. The IDE is geared towards touch based devices without keyboards, ranging from phones, over tablets, to large display screens. Programs can be edited and run on the device without an auxiliary PC. Transitioning between programming on one device, and continuing on another device is seamless. The web application also works offline.

TouchDevelop has been successfully applied to teaching introductory programming classes at the high-school level and at some college level for non-CS majors. For researchers, TouchDevelop provides a green-field platform to explore IDE and programming language design, as well as runtime techniques and distributed data storage abstractions.

In this talk, I will provide an overview of TouchDevelop from a language, IDE, and runtime perspective, while diving into some of the novel techniques enabled by our particular platform.

Categories and Subject Descriptors D.3.0 [*Programming Languages*]: General; K.3.2 [*Computers and Education*]: Computer and Information Science EducationComputer science education

Keywords Web IDE, Introductory programming, smart phone, tablet, touch-based entry

1. Introduction

TouchDevelop [1, 6] is a web-based integrated development environment (IDE) for writing small applications on touch-input devices such as smart phones and tablets. The initial motivation for TouchDevelop was to provide a creative outlet for today's masses of teenagers who carry around in their pocket a computer that is rougly 1000 times more powerful than the first computers many of our generation owned. What distinguishes our first computers from today's smart phones and tablets most is not necessarily the increased power, connectedness, graphics capabilities, sensors, etc, but the lack of simple programmability. Our first computers (for example a Commodore 20, or a Sinclair ZX81), presented the user with a Basic command prompt upon boot. Instead of enciting us to consume media on our devices, our first computers forced us to learn how to program. We don't advocate a return to 8-bit Basic, but today's devices simply make it too difficult to program them. They typically require a PC, an expensive developer registration, and extensive knowledge of how to build software. That is a difficult way to excite a teenager about programming. Our goal is to instead make learning to program as effortless as possible on today's devices.

2. Programming Language and Code Entry

What language do you use to teach someone to program and how do you balance simplicity against expressiveness? In TouchDevelop, we took a middle ground between making programming super easy for simple tasks, such as Scratch [5], and a fully expressive programming language such as C#. Initially, we tended towards a scripting language, such as Javascript, due its popularity with younger programmers. However, given our focus on touch-based entry of programs, we quickly realized that an untyped language does not provide us with enough context to provide intelligent suggestions to the programmer. The TouchDevelop language is thus typed. Types appear as declarations of procedure arguments and return values, as well as globals, and fields of records. Types for locals are inferred.

Whereas introductory programming might focus on a functional programming style to effectively strengthen the mathematical concepts, we found that programs for a phone are very much state based: reading the current sensor states, starting and stopping the playing of a song, displaying a picture. Our language thus ended up being a traditional imperative language with mutable globals, locals, procedures, loops, assignments, recursion etc.

Code entry in TouchDevelop is a mixture between structured editing at the statement/block level, and token entry at the expression level. Token entry is crucial for touch-based programming, as it avoids having to type names, keywords, etc, letter-by-letter and instead turns programming into a selection problem. We deemed using structured expressions too tedious due to the need to select what kind of expression to build before building it. Apart from a few infix operations, all operations are using an instance-method style selection form (`e->m`), which allows us to predict the set and order of properties/methods to propose based on the currently selected type of `e`. We are experimenting with various prediction methods to provide the list of most likely next tokens to the programmer.

3. Features

This section provides a short overview of some of the non-standard features of the TouchDevelop environment.

IDE as a Service All you need to connect to the TouchDevelop IDE is a modern web browser supporting HTML5 and Javascript. The TouchDevelop web-app stores scripts and application state both in local storage and in our cloud service. When connecting from a different device, your app state is downloaded from the

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
PEPM '14, January 20–21, 2014, San Diego, CA, USA.
Copyright is held by the owner/author(s).
ACM 978-1-4503-2619-3/14/01.
http://dx.doi.org/10.1145/2543728.2543746

cloud to provide the same environment as on all your other devices. Scripts can be published with one click in order to make them discoverable by other users. Additionally, art such as pictures and sounds can be stored and shared in the cloud as well.

Program Evolution All TouchDevelop programs are entered using our IDE, i.e., we don't give users a way to paste entire programs as text into our environment. Programs are created from a template, or forked from an existing program. As a result, we have quite a bit of insight into how programs evolve through user edits, which programs get forked by others, and what kind of modifications are typically done.

Besides the actual programs, the programming language and APIs of TouchDevelop also evolve over time. In order to not break existing users and programs, we carefully design changes to either be benign, or to provide a way to automatically upgrade existing programs. Since users only interact with their programs through our IDE, we can store arbitrary metadata alongside the program text and we have the freedom to upgrade their code to conform to API or language changes. This ability also relies on our IDE being a web-app in order for us to control versioning and upgrading of the IDE code itself.

Live UI Programming An expressive enough programming language quickly moves beyond simple console output and requires ways to build user interfaces and present data in appealing displays. TouchDevelop programmers build user interfaces using a page and box model that is quite different from traditional UI frameworks. Pages consist of vertical and horizontal boxes, similar to a TeX layout. These boxes are built, as in TeX, via the side-effect of a `boxed` statement. This approach allows using the entire programming language to construct layouts, such as conditionals, loops, recursion, etc. Furthermore, we strictly segregate layout code from normal state-changing code in order to perform automatic layout updates. In a running TouchDevelop program, the UI updates automatically based on changes to the state (through events etc.) without the programmer having to explicitly manage what happens on change. On each state change, we conceptually throw away the entire box layout and run the code again that builds it. Since that code is side-effect free (apart from layout side-effects), re-executing it does not affect program state, only UI state. Building UIs in this manner is surprisingly liberating, albeit not necessarily easy to teach.

The ability to rerun the layout code without affecting program state, allows our IDE to provide "live-programming", namely editing the layout code of a "running" program and seeing the changes to the UI immediately [3].

Cloud Data Useful applications on mobile devices rely increasingly on cloud storage in order to persist application specific data and to share it among multiple devices or users. Programming such cloud data sharing with the ensuing complications in terms of consistency and availability is beyond the ability of most professional programmers. One of our research agendas is to provide cloud shared data along with an easy consistency model to TouchDevelop programmers with the simiplicity of global variables. We have thus integrated recent progress on cloud datatypes [2] into the programming model of TouchDevelop, thereby enabling beginning programmers to write applications that would be highly non-trivial to write in traditional environments even by professionals.

Synthesis During expression building, users can enter search terms to find APIs and other program tokens. Synthesis extends this search beyond apis, by employing the cloud service to try to synthesize likely expressions or statements mapping to the search terms and variables available at the program point [4]. These synthesized expressions are presented like other search results and can be placed into the program with one click. The obvious drawback of search is that one has to type letter-by-letter. As a result, this mode is most useful when one does have a keyboard. Voice input is a possible future alternative, provided that the sound can be uploaded and recognized fast enough.

Publishing The TouchDevelop runtime is written in Javascript and TouchDevelop programs are compiled to a continuation passing form in Javascript. By default, our compilation targets browsers on mobile devices and PCs, but we can also target more specific environments, such as applications in traditional app stores. For example, TouchDevelop scripts can be exported as Windows Phone applications, or Windows 8 Store applications. In this mode, we simply package the runtime into an application that hosts a browser and then run the runtime in this hosted environment. Such platform specific hosting allows the runtime access to device features not traditionally exposed in HTML5 browsers, such as the users pictures and music stored on the device, or communication channels such as bluetooth.

Openness Finally, we want to make the vast data we have on TouchDevelop scripts and user interactions available to other researchers in the field. To this end, `touchdevelop.com` provides REST APIs for researchers to obtain parse trees and other information about scripts and our platform. These APIs allow third party researchers to write analyses, mine usage data, or write plug-ins to transform ASTs, e.g., for refactorings.

4. Conclusion

TouchDevelop is more than a toy. It has been used by over 140,000 users over the last three years and we see hundreds of active users daily. These users have published over 50,000 scripts. We invite other researchers to participate in the TouchDevelop experiment through the use of our open APIs, or to actively promote teaching introductory programming on TouchDevelop.

Acknowledgments

The TouchDevelop team consists of Michal Moskal, Nikolai Tillmann, Jonathan de Halleux, Sebastian Burckhardt, and myself.

References

[1] TouchDevelop Website. URL {`http://www.touchdevelop.com/`}.

[2] S. Burckhardt, M. Fähndrich, D. Leijen, and B. P. Wood. Cloud Types for Eventual Consistency. In *Proceedings of the 26th European Conference on Object-Oriented Programming (ECOOP)*, June 2012.

[3] S. Burckhardt, M. Fähndrich, P. de Halleux, J. Kato, S. McDirmid, M. Moskal, and N. Tillmann. It's Alive! Continuous Feedback in UI Programming. In *Proceedings of the 34th International ACM Conference on Programming Language Design and Implementation (PLDI)*, June 2013.

[4] V. Le, J. de Halleux, S. Gulwani, and Z. Su. Keyword Programming for TouchDevelop. In *Proceedings of the 11th International Conference on Mobile Systems, Applications, and Services*, 2013.

[5] J. Maloney, M. Resnick, N. Rusk, B. Silverman, and E. Eastmond. The scratch programming language and environment. *Trans. Comput. Educ.*, 10:16:1–16:15, November 2010.

[6] N. Tillmann, M. Moskal, J. de Halleux, and M. Fahndrich. Touchdevelop: programming cloud-connected mobile devices via touchscreen. In *Proceedings of the 10th SIGPLAN symposium on New ideas, new paradigms, and reflections on programming and software*, ONWARD '11, pages 49–60, 2011.