User Study on Implementation of Live Programming by Example

Background:

Live Programming by Example (LPBE) is a technology that is intended to be heavily interactive, with a focus not only in the raw code, but automatically generated outputs. For a given function, the developer is able to generate a sample output, and use that output to change the code by changing the desired output. LPBE has a large amount of potential with a wide range of potential use cases.

Programming by Example has been applied with great success in applications like Flash Fill for Microsoft Excel\(^1\). However, here the goal is not to replace programming knowledge, but to supplement and teach it. It is estimated that half of the time a programmer spends is spent understanding how legacy and other previously written code works alone\(^2\). By allowing modifications to the output to change the source code, programmers can better see and understand how the code they are working on works, and learn for the future. This can be applied to teaching people new syntax, as well as assisting locate changes in major variables, and aid modification of helper functions. Instead of telling them how to be coding, we show them how the code works, using examples and being hands on.

Currently, we have a working implementation of LPBE that allows basic JavaScript

---


functions to be modified and give examples as a package in the Atom text editor\(^3\).

However, this implementation is at a very early stage and has plenty of room to expand. Our goal moving forward with our LPBE Atom package was to figure out the next steps forwards to decide which direction to build upon it.

**Leading Questions:**

Before deciding to conduct a study upon this topic, it is important to decide which questions we are attempting to answer by performing it. After consulting with our research group, we decided to focus on a group of questions.

**R1:** LivePBE allows users to focus more on the abstract algorithms and functions in their programming instead of spending time learning legacy code and syntax in a traditional environment.

**R2:** By editing example outputs, users will either

1) Localize the problematic code in the source

2) Get the desired output immediately

3) Gain a better understanding of the code function through its variables.

**R3:** Live Programming by Example is simple to use and intuitive, requiring little to no time to learn, allowing users to spend their time on programing algorithms and functions instead of learning syntax and linting.

\(^3\) [https://github.com/santolucito/liveprogramming](https://github.com/santolucito/liveprogramming)
**R4:** Live PBE allows experienced programmers to adapt to new languages at least as quickly as existing approaches, such as searching the documentation.

**R5:** Live PBE helps new programmers understand how code structure behaves at least as well as existing approaches.

**Method:**

To conduct this study, our study conductor recruited eleven Yale College undergraduates to participate in a live, sit down interaction with our program. Our goal was to use a variety of users of multiple different skill levels at programming to get a sample of data to help guide our decisions.

Before using the LPBE package, each participant was asked to fill out a pre-study survey that would indicate their self defined proficiency with JavaScript as well as any other programming language, which they were asked to disclose.

During the study, each participant was given a short tutorial and explanation of the LPBE package, including what it was, creating a new output, modifying the outputs, and modifying the original functions. The participant was then given full control over the platform, and instructed to interact with it as they saw fit. They were encouraged to try to replicate the examples with new functions, as well as write their own original functions to test out the functionality. The study conductor was sitting alongside them and observing their behavior meanwhile, as well as to ensure that any bugs or crashes could be reset. Once the study participant had exhausted their tasks and curiosity with the platform, they were finished.

To conclude the study, the participants responded to a post-study survey, consisting of ranking the following statements: I found Live Programming by Example to be
helpful, I learned something new about JavaScript from Live Programming by Example. Using Live Programming by Example saved me time I would have spent learning syntax, I would use Live Programming by Example again for a new programming language, as well as asking for feedback in general and their definition of a helpful editor.

Results

A .csv file of the survey results is provided on this project website for individual download[^4].

As a whole, a majority of people would want to use LPBE for a new programming language, no matter their overall programming experience. Most users wanted to see a more clear tutorial as well. Clarity with the errors that appear and easy solutions to fix them in the editor were widely requested features.

Only those who were absolute beginners to programming were able to learn anything new about JavaScript from LPBE. However, there is no direct correlation between learning something new, and LPBE being helpful, as many found it to be a helpful program. Regarding syntax, the responses were very mixed, though there is a negative relationship between high general programming experience and learning syntax.

Overall, the general idea of a helpful editor was one that was able to seamlessly predict what the user wanted to happen without any intervention. Key concepts included saving time and energy that would need to be spent otherwise. Clarity and easy to understand instructions that assist in this were also common. Avoiding the need to consult online resources for help was ideal for some.

From study observation, it was noticed that those new to programming were able to understand the examples quickly, but did not know when syntax changes were needed, such as needed quotation marks to indicate strings.

[^4]: Please check the project website url for the active link.
Advanced developers were much more likely to try writing their own functions with more complicated concepts such as recursion, but were disappointed that they did not work as intended. Instead of modifying the recursive statements, it commented out the entire block and created a simple return statement.

All groups were overall mostly impressed by the program and found it fun and interesting. There was confusion in regards to updating the example outputs, with most forgetting to click to the other side of the editor to update them.

**Analysis and Conclusions:**

From this data set, it seems that the current best market for LPBE in its current form in our Atom package are people who are new to programming, but are not absolute novices to it. The people who seemed to be most engaged and interested with the platform were those who knew just enough to understand what a coding environment looked like and how to use it, but not enough to seamlessly synthesize and modify the functions as desired.

Of the main questions we wished to answer from our research, the main focuses in this study were to gain a better grasp on R2, R3, and R5 due to limitations in the ability of the package at this time. Currently, neither of them seem to be completely achieved as of now, but progress is being made. Given that LPBE is working with a commonly used text editor, it will be at the baseline as good as modern solutions in regards to code clarity, but much more information and guidance can be given.

Users wanted to see more instruction and interaction as a whole to use the platform than there would be in a bare IDE. There was a small learning curve that needed to be overcome, as there will not be a study conductor around to teach all new users how to use the platform. More direction in the README or a Wiki section in the repository would even
be a good way to get developers familiar with the GitHub platform a place to look for advice.

In its current state, new developers were able to learn the most from the platform, yet still wanted more hand holding in ensuring that their development felt efficient and simple to use. Taking the time to educate rather than fix without them knowing will ideally lead to better programming practices for these new to programming, and help them understand how the process works.

The results dictated that there is a future market for this type of development environment, yet there needs to be more fleshed out functionality and user experience considerations that should be fulfilled first. Many people found it to be helpful, indicating that live feedback will be a way to help streamline the development process.

I would encourage more similar studies to be done with this platform as it is expanded to get more user data. It is important to remember that these are subjective, self reported data points with a relatively small sample size, making this data more suggestive rather than definitive. In its current state, LPBE is very limited, and pigeonholed experienced users to more basic functions, but did not provide enough clear guidance to act as a tool for people new to programming. The directions needed to go for both of these markets require very different types of work, making the decision all the more important.

One suggestion I have for the future of this technology is to separate it from Atom and create a web app in which it can be accessed remotely, and without the barrier of another program for people to install. Many people are annoyed by the premise of having to download a new program, much less an add-on package for that program. Furthermore, many experienced developers already have an editor of choice that they are loyal to, and will not want the extra bloat of Atom just for one tool. Given Atom’s basis on Electron, a library that is built upon HTML, CSS and JavaScript, as it emulates a web page, this
should not take much extra effort\(^5\). A web app can allow many users to use the platform at a time seamlessly, and will push updates at your own accord. This can also allow another user study to be conducted remotely, which will allow a much larger sample of responses to be gathered.

Additionally, separating the “backend” logic of the LPBE infrastructure, the code directly related to the programming language itself, and the “frontend” interface of the package is ideal. This allows the code to be easily repurposed to a large number of different languages to aid in the high consensus that people would want to use this with a language they were new to. As this project is open source on a GitHub repository, it can also lead to allowing people to independently creating these expansions for any language they want. Currently, a branch made for Haskell is being developed, and this could be accelerated with this separation\(^6\).

If the decision is to target novice developers, the presentation of the package needs to be improved. This is already being worked on by this team in the "uiupdate" branch in the GitHub repository this is being hosted on\(^7\). But additional tutorial pop-ups, documentation, and perhaps even guided samples or video would be immensely helpful in aiding their ability to interact without human intervention. Additionally, helpful error checking pop-ups for common issues such as syntax errors to provide live feedback for their interaction should help prevent common issues that were observed in the study.

If the decision is to target more experienced developers, the algorithms to create LPBE need to be expanded further to allow them to work with more complication function that utilize more complicated statements they are actively using. Additionally providing a more transparent explanation of how the algorithm makes the choices it does in regards to

---

\(^5\) [https://electronjs.org/](https://electronjs.org/)

\(^6\) [https://github.com/santolucito/liveprogramming/tree/haskell](https://github.com/santolucito/liveprogramming/tree/haskell)

\(^7\) [https://github.com/santolucito/liveprogramming/tree/uiupdate](https://github.com/santolucito/liveprogramming/tree/uiupdate)
function modification. An example that a more experienced programmer wanted to see was the ability to have the examples become uncommented when a created example was removed to revert back to the original code.

In conclusion, Live Programming by Example seems to be a success in captivating people’s interest, but still needs work to encourage people to actively use it. To progress further, expansion must be made in the clarity and functionality of the platform. Those that seem to be the best fit to focus towards are novice programmers who are able to learn more clearly on function interaction and proper structure by learning from example. Providing an accessible, easy, and fun way to use the platform, it can help people become more invested in learning how to code by providing live feedback.

Acknowledgments:

I would love to thank my advisor Ruzica Piskac for inviting me into her team to aid her in the research of this amazing technology. She has been extremely supportive the entire way through, and has been extremely helpful and welcoming to me as a novice researcher. I would also like to thank Mark Santolucito and William Hallahan, who were my main points of contact with the project and aided me getting everything set up and formally conducted. Our conversations were very fruitful, and I hope that this report is able to aid them with their goals for the Live Programming by Example experience. I would finally like to thank all users who aided me on the Atom help forums and other internet communities as I was learning Atom package building for the first time.