Dynamic Tutorials: Creating Readily-Available Worked Examples for Novice Programmers

Modern professional web development relies heavily on front-end frameworks, such as AngularJS, and techniques designed to augment and simplify the capabilities of native web technologies. Although these tools and techniques provide helpful abstractions for experienced developers, their complex interaction protocols present a steep learning curve. Learners must not only understand *how* to use features of a particular technology, but also *when* to apply usage patterns and techniques to specific cases. For example, an intermediate web developer interested in copying a particular Facebook feature might investigate the underlying source code. However, professional websites frequently obfuscate their production code, use highly-complex application architectures, and generally provide little readability for learners unfamiliar with the code base. Consequently, intermediate learners find it difficult to connect a website's source code to outcomes on the page -- a *knowledge gap* preventing the accumulation of professional-level techniques. Drawing from professional examples, we will help intermediate web developers fill their knowledge gaps so they are able to implements certain features and learn the best design patterns that are used by professionals. We propose to create a system called Dynamic Tutorials, where we will create tutorials about front-end development techniques, such as grid systems and the ideas of models, views, and controllers, using professional examples and present them in a two different tutorials. Both tutorials will serve to demonstrate the same high-level concept or programming "design pattern," but the individual tutorials will implement these concepts in different concrete ways. Using our application, Dynamic Tutorials, we will then conduct user studies to determine the effectiveness of our learning application. We will research whether multiple examples teaching core concepts of front-end frameworks will facilitate analogical encoding. Both of us are deeply interested in researching the intersections of computer science and learning sciences; as teaching assistants and human-computer interaction researchers, Dynamic Tutorials present a niche opportunity to draw from our interdisciplinary backgrounds.

While existing online tutorials demonstrate the "how to do," most do not provide the "when to do" intuition necessary for effective knowledge mapping to related contexts. This past fall and winter, we conducted preliminary needfinding interviews with 10 programmers ranging from novice to intermediate. In these interviews, learners repeatedly expressed frustration with the overly-focused nature of existing web tutorials. They found the example scenarios too contrived, and noted that instructions tended to focus on low-level tasks instead of the generalizable reasoning behind each operation. Even when tutorials evade these pitfalls, learners tend to encode information in a context-specific manner when learning from a single case (Gentner & Rattermann, 1991). Conversely, explicitly comparing two different examples facilitates *analogical encoding*. Analogical encoding describes the promotion of structural transfer via comparison of two partially-understood examples. Multiple examples of the same principle and technique make common relational structures more salient, and can facilitate deriving a schema, which in turn enables the learner to subsequently apply the acquired knowledge to similar problems (Gentner, Loewenstein, & Thompson, 2004; Gick & Holyoak, 1983; Catrambone & Holyoak, 1989).

Although these observations are well-documented in the CS+LS (computer science and learning sciences) field, our proposed work builds upon the existing literature by focusing on the needs of *intermediate* learners specifically. Interactive web platforms such as Codeacademy provide a rich array of learning environments for novice learners with zero prior programming experience. However, once a learner has developed the fundamentals, next-step resources become far scarcer. Indeed, the vast majority of existing research in the CS+LS domain focuses

on novice programmers with no prior technical background. Most of the state-of-the-art learning affordances and findings have been designed with a very different user class in mind, and translate poorly to filling the knowledge gap for intermediate programmers. Prior research has documented learning behaviors typical of intermediate programmers, and shown the benefits of integrating example code "snippets" into their workflows (Brandt et al., 2009; 2010). While these contributions are preliminary findings, they provide the basis for our current proposal. We believe our work can augment the CS+LS domain by specifically targeting intermediate learners, and further understanding their unique needs vis-a-vis the widely-studied class of novice learners.

Dynamic Tutorials can roughly be thought of as "educational faceplates" for the internet, and differ from existing tutorials in three ways. *Web-sourcing of example material.* Currently, tutorial authors must build their own sandbox environments to demonstrate a general concept. For instance, someone interested in authoring a tutorial for auto-completing search bars must create a mock website with a "dummy" search bar and mock data for autocomplete. This process is time-consuming, and often results in a contrived and narrow product (in turn promoting the learner's contrived mental schema). Harnessing the vast body of existing websites to populate worked examples would provide a huge amount of learning material. For example, learners are already familiar with sites they use, such as Facebook, and using Facebook's source code as part of the tutorial, provides context to the learner. *Repetition of concepts to promote analogical encoding.* Existing tutorials demonstrate one principle in one way. However, learners empirically develop more robust knowledge transfer when comparing *two or more* examples of the same technique. Because other tutorials use different approaches and technology stacks, learners cannot easily reinforce their newly-acquired knowledge within a different context. One Dynamic Tutorial could theoretically be used to generate many different examples of the programming technique in question. Learning how Facebook is creating a search feature and learning how Twitter creates a search feature, provides learners with two different examples in context to better encode what they are learning. *Authentic learning through real-world examples.* Live examples may be more relevant to the learner's aspirations and interests, facilitating authentic learning (Shafer and Resnick, 2003). Ideally, the user could generate custom Dynamic Tutorials, thereby contextualizing a particular high- or medium-level programming plan to a website of their choice. Facebook examples will provide more context to a learner v.s. an isolated example from a tutorial.

Methodology

We aim to answer the following research questions:

1. Do real-world worked examples of programming techniques increase learners' rates of knowledge transfer, compared with sandboxed tutorials?
2. Do multiple real-world examples increase learners' rates of knowledge transfer, compared with only one real-world example of a programming recipe?

We define "rate of knowledge transfer" to mean "the rate at which learners successfully apply the general recipe to a new case."

We plan to spend 4-6 weeks developing the Dynamic Tutorials system using JavaScript-based technologies. During this time, we will also iteratively prototype and refine the user interface for the system, based on user testing with a group of learners. We will begin user testing this coming week, but will need funding to compensate learners for long-term

participation from our recruited group of learners in our design process. We will recruit 5-6 long-term participants will have experience with HTML and basic JavaScript, but minimal to no prior experience with AngularJS. They will be testing our interface. We will iterate upon our system with the help of our long-term participants who will help test our system for usability and intuitive design.

In the following 4 weeks after development has concluded, we will create a set of Dynamic Tutorials for a programming recipe using web techniques and JavaScript framework specific techniques, and find a set of comparable traditional tutorials. [Student One] will develop the back-end system to the tutorial, while [Student Two] will focus on creating and designing the interface learners will interact with. Back-end tasks, which will be completed by [Student One], will include setting up a server, creating hooks to search and retrieve content from websites. The interface, built by [Student Two], will include thinking of the design affordances our tutorial will have and how to layout the features. To have a statistically-valid study, we will need to test with at least 25-30 learners; each learner requires a hour of a highly involved study. By having two people we will be able to get statistically-valid results by conducting the study with more learners. We will meet weekly with Dr. [Faculty Mentor] to plan the work throughout the next two quarters. Once we have a interface, we will conduct a user study. We will conduct a user study with 25-30 new undergraduates, who have the same background as our long-term participants. Undergraduates will be recruited from several sources, including EECS 111, EECS 330, and web development related student groups.

Our study will have four conditions: the participant is given one real-world Dynamic Tutorial, one sandbox traditional tutorial, two real-world Dynamic Tutorials, or two sandbox traditional tutorials. Participants will be randomly assigned to a condition.

Each participant will work through their provided tutorial(s), and then complete an exercise consisting of a blank version of the Dynamic Tutorial and a hypothetical context. We will evaluate how the users are able to translate skills learned from the tutorial to a mock scenario where they will be building what they've learned from their tutorial. We will also be conducting observations during the study examining how the participants navigate the different tutorial conditions. We will evaluate whether or not learners are able to complete tasks in the mock scenario. During the observations, we will be focusing on how users are interacting with the tutorial. We will also get subjective feedback from users, asking them to rate how they liked the tutorial, and if they felt like they learned better from the Dynamic Tutorial. We will analyze the subjective and objective evaluations from the different user conditions to see if Dynamic Tutorials have better learning outcomes for users, and if users felt more empowered and enjoyed using Dynamic Tutorials more.

We both have previous experience in conducting Human-Computer Interaction research and understanding Human-Computer Interaction principles with Dr. [Faculty Mentor]. [Student One] is currently the TA for EECS 330: Human-Computer Interaction. [Student One] has had experience deploying app systems during previous internships at Groupon and AT&T and have experience through her interfaces classes at Northwestern. [Student Two] has built many front-end applications with various web frameworks. As a TA for EECS 111: Fundamentals of Computer Programming, she is familiar with the learning needs of novice and intermediate-level programmers. We will submit the results of our research in the form of a conference paper at the end of the quarter or next quarter. This experience will support our long-term goals of attending graduate Computer Science programs.

Works Cited

Brandt, J., Dontcheva, M., Weskamp, M., and Klemmer, S.R. Example-centric programming: integrating web search into the development environment. Proceedings of CHI 2010, ACM (2010), 513-522.

Brandt, J., Guo, P.J., Lewenstein, J., Dontcheva, M., and Klemmer, S.R. Two studies of opportunistic programming: interleaving web foraging, learning, and writing code. Proceedings of CHI 2009, ACM (2009), 1589-1598.

Catrambone, R. & Holyoak, K. J. (1989) Overcoming contextual limitations on problem-solving transfer. Journal of Experimental Psychology 15:6, 1147-1156.

Gentner, D., Loewenstein, J., & Thompson, L. (2004). Analogical encoding: Facilitating knowledge transfer and integration. Proceedings of the Twenty-Sixth Annual Conference of the Cognitive Science Society, 452-457.

Gentner, D., & Rattermann, M. J. (1991). Language and the career of similarity. In S. A. Gelman & J. P. Byrnes (Eds.), Perspectives on language and thought: Interrelations in development (pp. 225-277). London: Cambridge University Press.

Gick, M. L., & Holyoak, K. J. (1982). Schema induction and analogical transfer. Cognitive Psychology 15:1, 1-38.

Shaffer, D. W., & Resnick, M. (1999). "Thick" Authenticity: New Media and Authentic Learning. Journal of Interactive Learning Research 10:2, 195-215.