world software development, we can incorporate software engineering best practices, introduce students to continuous integration in practice, and provide formative feedback to students throughout the software development lifecycle. We found that 76% or more of students in each of the classes that deploy our framework reported that using Jenkins increased their productivity, and that 84% or more of students in each of the classes reported that using Jenkins increased their code quality.

**ACM Categories & Descriptors:** K.3.1 [Computers and Education] Computer-assisted instruction (CAI)

**Keywords:** software engineering skills; continuous integration; automated evaluation and feedback

**DOI:** http://dx.doi.org/10.1145/2676723.2691921

## A Bottom-Up Approach to Teaching Server-Side Web Development Skills

Ariel Ortiz, *Tecnológico de Monterrey, Campus Estado de México*
Contact: ariel.ortiz@itesm.mx

When dealing with the topic of back-end programming many CS web development courses typically focus on how to use a popular web framework, for example Spring MVC or Ruby on Rails. The problem with this approach is that students will most likely end up using some other different framework or technology if ever they decide to become professional web developers. Our students need to learn concepts and skills that serve as a foundation to learn whatever different technologies are used now or happen to appear in the future. This poster presents the author's experience on using a bottom-up approach to teach the fundamental aspects of how the HTTP protocol works, and how this knowledge can be used to get a deep understanding of the inner workings of the web by building a simple yet complete server-side web framework. Using Node.js as the development platform, students are able to take TCP sockets as the building blocks for higher-level web abstractions. This approach allows covering a variety of specific topics that are essential for a professional web developer: request and response structure and headers, HTTP methods, form processing, cookies and sessions, text encodings, MVC software architectural pattern, database integration using ORM (Object-Relational Mapping), REST (Representational State Transfer) architecture, security issues (HTTPS protocol, common web vulnerabilities), and client-side integration using AJAX (Asynchronous JavaScript and XML). Anecdotal evidence shows that students with this knowledge repertoire are better suited for learning, using and debugging new and existing web technologies.

**ACM Categories & Descriptors:** K.3.2 Computer and Information Science Education; C.2.4 Distributed Systems

**Keywords:** Web development; Server-side programming; JavaScript; Node.js

**DOI:** http://dx.doi.org/10.1145/2676723.2691922


*SIGCSE'15*, March 4–7, 2015, Kansas City, MO, USA.
ACM 978-1-4503-2966-8/15/03.

## Teaching Text-based Programming in a Blocks-based World

David Weintrop, *Northwestern University*
Uri Wilensky, *Northwestern University*
Jennifer Rosceo, *Lane Tech College Prep*
Daniel Law, *Lane Tech College Prep*
Contact: dweintrop@u.northwestern.edu

This poster presents an environment and set of pedagogical strategies designed to explore how best to use blocks-based programming tools to prepare learners for future, text-based programming languages. Starting with the snap! programming environment, we added the capability to view the JavaScript equivalent of any blocks-based script authored inside the environment. Additionally, when students define behaviors for new blocks, they do so in JavaScript. This makes it possible to compose blocks-based scripts alongside text-based JavaScript programs and have the two run side-by-side. This environment was used during the first 5-weeks of an introductory programming class at the high school level as part of a quasi-experimental study investigating the relationship between programming modality and emerging student understanding. Teachers of the course used the blocks/text hybrid features in various ways to support learners developing an understanding of programming concepts and laying the foundation for future text-based instruction. These strategies included having students compose programs with graphical blocks then view the equivalent JavaScript, prompting class discussion on similarities and differences between the two modalities; having students write pseudocode for their blocks-based programs before comparing the pseudocode to the JavaScript; and finally, having students implement their algorithm directly in JavaScript, using blocks as a resource to reference proper syntax.

**ACM Categories & Descriptors:** D.1.7 Visual Programming. K.3.2 Computer and Information Science Education

**Keywords:** Introductory Programming Environments; Blocks-based Programming; High School Computer Science

**DOI:** http://dx.doi.org/10.1145/2676723.2691923

## Summer Programming Boot Camp: A Strategy For Retaining Women In IT

Sonal Dekhane, *Georgia Gwinnett College*
Kristine Nagel, *Georgia Gwinnett College*
Nannette Napier, *Georgia Gwinnett College*
Contact: sdekhane@ggc.edu

This project addresses the issue of retaining women in Information Technology (IT) at an open access institution. To meet the goal of retention, we focused on supporting students' learning and mentoring needs. Female IT majors and minors were recruited and participated in a weeklong summer boot camp. At the boot camp they participated in Java programming sessions, various professional development and peer mentoring sessions and a field trip. These activities were aimed to increase not only the participants' knowledge of programming, but they were also designed to increase the participants' confidence, their knowledge of IT careers and fields of research and most importantly to form a community of support. Initial data collection shows that 61% of participants enrolled in a programming course following the boot camp, which is a crucial step in retention of majors and minors. Authors have also experienced an increase in the participants'