The Effectiveness of Online Coding Courses

A Research Paper submitted to the Department of Engineering and Society

Presented to the Faculty of the School of Engineering and Applied Science

University of Virginia • Charlottesville, Virginia

In Partial Fulfillment of the Requirements for the Degree

Bachelor of Science, School of Engineering

Mohammed Alwosaibi

Spring 2025

Advisor

Joshua Earle, Department of Engineering and Society

Introduction

Most students join online coding courses to acquire applicable coding skills, which translate to jobs, improved problem-solving abilities, or creative independence. Courses like Coursera, Udemy, and Codecademy have made learning easy and convenient, but students go through tutorials without becoming proficient programmers on their own. It is also called "tutorial hell," where students depend excessively on step-by-step tutorials and fail to master problem-solving capabilities. In this paper, I ask whether online coding platforms foster independence or teach students to follow instructions.

The effectiveness of online courses varies widely between platforms. Some emphasize videos, others offer interactive challenges, and a few provide real-world projects or peer collaboration. However, many students drop out or feel unprepared when the material becomes overwhelming or disconnected from practical use. Structured lessons may explain concepts, but students often forget key ideas without applying them or lose interest. This variation in format raises an important question: Which course design best facilitates long-term participation and skill learning?

The biggest flaw of most online sites is the lack of real coding practice and constructive criticism. While in conventional schools, students can ask questions and interact with other students, most online courses use computerized quizzes and pre-written assignments. These may test syntax but not design, debugging, or application-building skills. The most effective way to become a strong programmer is through building complete projects, facing bugs, and solving open-ended problems, elements often missing in purely tutorial-driven platforms.

Educational theory provides insight into why many online courses fall short. Constructivist learning emphasizes hands-on experience and discovery, which is especially critical in programming (Reese, 2011). Cognitive load theory explains why students struggle when overwhelmed with content without enough practice opportunities (Van Merriënboer & Krammer, 1987). From an STS perspective, course design often relies on economic and institutional goals, like maximizing enrollment or subscription retention, rather than pure educational value, demonstrating the effect of the social construction of technology (Pinch & Bijker, 1984). These structural influences can undermine the development of real problem-solving ability.

With so many people unsure which platform to trust, understanding what makes an online course effective is more important than ever. Course providers must refine their methods to help students move from passive tutorial-following to active coding (Kim & Ko, 2017). By analyzing popular online coding platforms and how they align with practical teaching principles, I seek to uncover which environments truly prepare learners to become independent developers and which may hinder that growth.

Methods

I employ a mixed-method approach in this paper to investigate how online coding classes facilitate the attainment of independent coding skills. The study begins with selecting courses on popular websites like Coursera, Udemy, Codecademy, edX, and freeCodeCamp. I chose these sites because they are well-known and offer different teaching styles, from video education to interactive practices and project work. We want to analyze users' highly rated and well-reviewed courses to get a balanced representation. We incorporate different formats, such as instructional videos, guided exercises, real-world projects, and community support.

This study assesses courses by examining key factors that influence learning outcomes, such as interactivity, hands-on projects, opportunities for feedback, and community involvement. Previous research in programming education indicates that guided learning through active engagement, where students practice material instead of passively consuming it, yields better long-term skill retention (Merrill et al., 1995). Courses that rely exclusively on video tutorials often make students overly dependent on them. On the other hand, hands-on coding, problem-based learning, and cooperative problem-solving foster greater autonomy for students. The two categories are determined via a qualitative content analysis of these courses based on their instructional design. It scans each course to assess its integration of practice and theory and whether students solve problems independently or from directions. The analysis finds two categories: tutorial-based courses that provide step-by-step instructions and project-based courses that encourage creating applications from scratch. It also considers courses that include interactive coding exercises that are fun for students but may not adequately prepare them for independent work.

A quantitative assessment utilizes student responses and completion rates to enhance this qualitative evaluation. Reviews of Udemy and Coursera often highlight challenges in applying learned skills to practical scenarios outside of structured exercises (Kim & Ko, 2017). Many online coding courses emphasize syntax and isolated tasks but fall short of helping students synthesize their skills into comprehensive projects. This research analyzes patterns in student feedback to determine how frequently learners report difficulties with coding independently after completing their courses. In addition, completion rates are an indirect indicator of engagement.

The literature supports that project-based, hands-on learning courses retain students at a higher rate because students tend to be more interested if they see progress in their work (Yeom et al., 2022). In addition, a case study follows a learner's journey from tutorial dependence to coding independence. It discusses their challenges with online coding lessons and reliance on pre-formatted content. Ultimately, the student solves these issues by engaging in coding forums and practicing independently, illustrating the tenets of the constructivist theory of learning, whereby knowledge is constructed actively by solving problems (Bijker et al., 1987).

Practical Learning

freeCodeCamp is better than conventional learning due to applied learning incorporating in-context coding exercises and comprehensive projects that structure its curriculum. From the start, students are engaged in actual problem-solving and mobile app development, fostering independent practice rather than passive content consumption. This project-based approach equips students with hands-on experience directly relevant to development work.

Udemy primarily relies on pre-recorded video lectures, and while some instructors integrate coding exercises, many emphasize students simply copying code from the lectures. This limits opportunities for independent problem-solving, resulting in students gaining theoretical knowledge without developing strong independent coding skills unless they seek additional challenges from outside sources.

Coursera provides a more interactive experience than Udemy. Most courses include auto-graded exercises and hands-on programming assignments that offer instant feedback. Additionally, some courses have capstone projects that encourage practical application.

However, the level of hands-on application differs significantly among courses, depending on the instructor or institution.

Odin Project is profoundly committed to project-based, real-world learning. It eschews dusty video lectures, giving students the challenge of reading documentation, concept application to projects, and debugging independently. This method makes a student more self-reliant and sharpens useful debugging and research skills that real developers need.

Codecademy balances instruction and practice, providing an in-browser development environment that allows students to write and execute code directly with immediate feedback. However, although it offers real-world projects in its paid tier, the overall number of substantial, portfolio-worthy projects is less compared to platforms like freeCodeCamp and The Odin Project.

Cost and Accessibility

freeCodeCamp is free and grants full access to its curriculum, including certification. This makes it one of the most cost-effective platforms for students with tight budgets. All learning materials are available online without paywalls, offering comprehensive programming education at no cost.

Udemy operates on a per-course pricing model, requiring a one-time payment for each course. While individual classes may be discounted to as low as $10 to $20, the overall expenses can accumulate when students need to purchase multiple courses to cover various topics.

Although Udemy may appear cheaper upfront than a subscription model, long-term costs can be less predictable.

Coursera offers some free content; however, most graded assignments and certificates necessitate a monthly subscription fee. Specializations typically range from $39 to $79 per month. While financial aid is offered, the subscription costs can challenge those who may require additional learning time or wish to progress slowly.

The Odin Project is entirely free, including every lesson and project. It utilizes free, readily available materials, so it is very cheap. However, occasionally, students may need additional readings or resources of their own will, some of which can be costly if the students choose to purchase them.

Codecademy operates on a subscription basis, with limited free content and a paid-for Pro version costing around $40 monthly. The Pro version provides career courses, real projects, and quizzes, and the repeat fees can be off-putting for those who prefer cheaper pay-as-you-go options.

Job Preparation

freeCodeCamp effectively equips learners for professional careers by emphasizing practical, real-world projects that can be showcased in a portfolio. Its web development and data science certification tracks enable students to acquire in-demand skills employers value. Although its certificates are not officially accredited, most graduates have found developer positions by leveraging their projects.

Udemy, however, offers a highly unpredictable experience in terms of job readiness, relying heavily on the teacher. Some courses provide valuable guidance on portfolio creation and interview techniques, while others deliver content without structured support. Consequently, students may need to supplement Udemy courses with additional resources to adequately prepare for their careers.

Coursera is particularly well-suited for those seeking formal credentials and employment preparation. It comprises courses by reputable organizations such as Google, Meta, and prestigious universities, which provide established certificates and job-focused content, generally involving capstone projects and peer-reviewed assignments. Job placement and mentorship support are limited, so intrinsic motivation is critical in completing the course.

The Odin Project places heavy emphasis on preparing students for real-world development scenarios. It offers in-depth training in Git, debugging, deployment, and best practices in software engineering, capped off with a full-stack portfolio. Its focus on solo problem-solving mirrors the scenarios that professional developers really face.

Codecademy provides structured career tracks and pathways for full-stack developer and data scientist roles. While these pathways cover essential technical skills, they tend to involve fewer extensive projects necessary for constructing a robust portfolio. Though Codecademy’s Pro model includes career counseling and interview preparation, it does not offer formal accreditation or job placement assistance.

Support and Community Engagement

FreeCodeCamp has one of the most engaged coding communities, which includes forums, Discord servers, and GitHub discussions through which students can reach out to their peers and obtain feedback. The peer support mechanism is especially effective for new students who thrive on advice and support throughout their learning journey.

On the other hand, Udemy has minimal community support. Even though every course has a Q&A feature, the responses are subject to the availability of instructors, and thus there is limited peer interaction. This could be a weakness for students who perform better with continuous feedback and peer encouragement.

Coursera provides discussion boards in its courses, where students can ask questions, discuss ideas, and review peers' assignments. Some courses also offer mentorship or networking opportunities; however, the level of interaction can vary depending on the course provider. The platform isn't as known for its robust, interactive community.

The Odin Project maintains a strong community presence through its Discord server, where students collaborate on projects, provide feedback, and receive assistance from experienced developers. This fosters a sense of community and approaches a collaborative study environment.

Codecademy features forums and community elements, but it lacks an emphasis on peer interaction. While the platform offers structured support and instant feedback, learners seeking more engaged interaction or mentorship may find it less beneficial than freeCodeCamp or The Odin Project.

Optimal Platform Selection Analysis

Before proceeding further into this analysis, one must acknowledge a universal principle: these sites are meant chiefly for beginners. Intermediate or expert developers will likely be better served by developing projects, reading manuals, or seeking inspiration from real-world problems. Whether the goal is to freelance, land a job, create content, launch an agency, or develop a SaaS product, learning web development and becoming proficient is the initial objective. The learner's long-term ambition does not affect how effective a given platform is during the beginner phase. Therefore, this section focuses on which platform offers the best return on investment, whether that investment is time or money while aligning with the student's learning preferences and helping them transition from absolute beginner to independent developer.

freeCodeCamp has solidified itself as a go-to for aspiring developers, especially with the launch of its latest Full Stack Developer Path. Its most significant strength is likely to have a linear, straightforward format. Beginners do not have to worry about learning what to learn next because the site maintainers have organized the course logically and step-by-step. The browser-based, interactive coding environment obviates the obstacle of in-house tool setup, which is usually annoying for novices (Handur et al., 2016). The other primary advantage is the vast international community of the platform. When a student gets stuck, some other student likely has run into and solved the same issue. However, freeCodeCamp's material occasionally overwhelms users with extensive videos and concentrated content. According to cognitive load theory, an overload of passive information without sufficient practice can hinder long-term retention (Mayer, 1981; Van Merriënboer & Krammer, 1987). Nevertheless, with its strict instruction, interactive coding classes, and successful peer assistance, freeCodeCamp is among the top free options for beginners looking to become professional developers.

Udemy is affordable and has a wide range of materials. With courses frequently discounted to between $10 to $20, students can experiment with various subjects. It helps fill unaddressed gaps through platforms like freeCodeCamp or The Odin Project, such as mobile app development or niche frameworks (Kim & Ko, 2017). Its flexibility makes it an excellent addition, allowing students to experiment with tangential interests. Its overallist approach does so at the cost of depth. Many courses consist of small, disconnected projects or exercises filled with boilerplate code, limiting opportunities for critical thinking and conceptual integration (Reese, 2011). Students often finish a course without being able to synthesize or apply the material independently. While Udemy serves well for quickly acquiring surface-level knowledge, it frequently lacks the scaffolding necessary for deep learning and real-world readiness.

Students often praise Coursera's academic partnerships with institutions like Google, Meta, and leading universities. These collaborations result in high-quality content and certificates that can carry professional credibility. The well-structured courses typically include graded assignments, peer-reviewed work, and structured feedback mechanisms (Anderson et al., 1989). However, Coursera can feel overly formal and time-consuming for students who specifically want to focus on web development. Some tracks require hundreds of hours to complete, which can overwhelm beginners and trigger issues related to cognitive overload, especially when hands-on practice is limited (Van Merriënboer & Krammer, 1987). Also, Coursera's subscription-for-money approach can prove expensive for slower-learning students. While it offers depth and scholarship luster, Coursera is best for those who like structured, collegiate environments and do not mind giving a lot of time and money.

Odin Project provides an intense and organized path to web development skills. It is free and open-source, and it stands out by requiring students to set up their local development environments, which mirrors real-world developer workflows and helps learners become comfortable with professional tools like Git and VS Code (Handur et al., 2016). The platform includes a wealth of reading material and practical projects encouraging problem-solving and independent thinking. However, the almost singular emphasis on text is overwhelming. Students indicate that they are covering several lessons without ever having to put down a single line of code, potentially leading to students checking out if the learner is not motivated or patient (Mayer, 1981). Despite this, for learners who prefer depth and are comfortable with theory-first instruction, The Odin Project offers one of the most comprehensive and job-aligned learning experiences available for free.

Codecademy is known for its accessible and interactive design, featuring an in-browser code editor and instant feedback. This setup is ideal for reducing technical barriers, allowing beginners to focus on learning concepts rather than configuring environments. However, its broad array of learning paths can dilute the platform's focus on web development. Certifications offered by Codecademy cost extra, and it is not very well known that employers recognize them. Moreover, while the platform includes real-world projects in its Pro plan, the quantity and depth of these projects are generally insufficient to stand out in a competitive job market (Kim & Ko, 2017). As a result, Codecademy works well as an introduction to coding concepts but falls short in helping students transition to building complete applications or developing a professional-grade portfolio.

Structure is essential in online coding education. A curriculum that balances conceptual instruction with real application helps learners translate knowledge into action. While depth is important, it must be tempered with practical reinforcement to avoid cognitive overload (Mayer, 1981; Merrill et al., 1995). Although certifications from platforms like Coursera can strengthen a resume, employers continue to place more value on demonstrable skills through personal or collaborative projects (Kim & Ko, 2017; Malik & Zhu, 2023). Here and in the outcome, The Odin Project and freeCodeCamp are the best platforms for new web developers. Both websites are superior in different aspects: The Odin Project is optimal for readers who prefer extensive reading and a professional setup process, and freeCodeCamp for readers who work better with organized challenges and practice immediately. Ultimately, the best platform depends on the student's preferred learning style, but both paths offer the tools needed to become a self-sufficient and capable developer.

Discussion

The outcome of this study provides a better idea about the situation of online coding tutorials and whether they prove helpful in letting individuals code alone. Online coding tutorials are at hand and are easy to approach but vary based on the learning pattern. This determines how much the students keep going on coding on their own. In this chapter, the implications of the findings are outlined, course improvement recommendations are provided, and areas for further research are suggested. Most coding courses online can convey introductory notions but fail to prepare learners for real-world programming tasks (Kim & Ko, 2017). Video tutorial-based and led instruction classes lead to passive learning whereby learners undergo instructions without being actively engaged. Learners then get stuck in "tutorial hell," where they can copy code but cannot independently apply concepts (Merrill et al., 1995). This is concerning because more people increasingly rely on online learning for technical skills.

This has profound implications for software development and job preparedness. Employers want people who possess the capability to resolve things on their own, code competently, and work with minimal micromanaging. However, if online tutorials keep producing students who understand how to work tutorials, then this will be a problem. Then, the industry can face a skills gap where candidates lack the practical experience and problem-solving abilities required in professional environments. There is a gap between education and industry needs. Providers of online courses must improve their teaching practices to meet real-world programming skills (Malik & Zhu, 2023).

Adding more hands-on, project-based learning can help address this issue. This study shows that practical projects from freeCodeCamp help learners develop better coding skills and independence. These real-world tasks encourage critical thinking and problem-solving. Courses that ask students to create complete applications help them move beyond memorization and develop skills to tackle complex coding challenges. Online courses can also improve learning through teamwork and community involvement. Coursera and edX demonstrate how social learning improves online education. Collaborating allows students to share knowledge, learn from one another, and enhance problem-solving skills, mimicking some advantages of traditional classrooms.

Peer collaboration needs a clear structure and active moderation. The study found that many course community features are underused or poorly managed, leading to low engagement.

Course providers should build supportive communities where learners want to participate, such as hiring mentors to lead discussions and create opportunities for group projects or coding competitions. Feedback is essential in online coding courses. Automated checks help students find mistakes, but the key point is that they do not always prepare individuals for real-life situations where they will need these skills. Courses should provide personalized feedback, such as code reviews. Reflective activities can also improve problem-solving skills and promote independent learning.

The study shows that free platforms like freeCodeCamp, which prioritize education, are more effective for developing coding skills than some subscription services. This focus on practical education suggests that financial incentives in online education can shape course design in ways that prioritize engagement metrics, such as course completion rates or subscription renewals, over meaningful learning experiences. Course providers must balance financial health with the responsibility to deliver quality education that prepares learners for successful careers.

Online learning platforms that turn education into a product raise concerns about access and how well students learn over time. Many of these platforms use a subscription model, encouraging users to keep paying instead of focusing on learning skills (Xinogalos, 2016). This money-driven approach affects the incentives these courses and their creators rely on. It often focuses more on keeping students engaged than helping them truly understand and remember the material. This situation relates to ongoing discussions in the Science and Technology Studies section about how money influences technology and affects how people access and use educational resources.

freeCodeCamp is a non-profit learning platform offering free learning material or for profit. The fact that there is no profit highlights the significant issues around how profit affects online learning. Online coding courses might be easy to do, but they do not necessarily pay enough attention to what happens to students. This concept shows that it is crucial to include perspectives from science, technology, and society when evaluating the long-term effects of these courses.

The instructors should embrace a mixed approach with the integration of diverse modes of teaching to reinforce online coding training. The courses can begin with guided walkthroughs, move on to drill and practice exercises, and include full-blown projects. The projects will help the students build confidence and implement theory in real-life applications. It is a necessity that we find ways to make learning more appealing and find ways to apply the knowledge we gain. Future research could look into new technologies like virtual reality or gamification. It is our task to use artificial intelligence, machine learning, and the power of modern technology to find better ways of teaching and informing the newer generations of the vast amounts of knowledge we have accumulated.

Personalized learning is essential. Adaptive technologies can generate adaptive learning routes. Teachers must be trained to produce actionable online courses and emphasize active learning and collaboration among students. Teacher mentoring can enhance online coding education and better prepare students for careers in technology. Online coding courses can be helpful, but their success depends on their structure. Programs that include hands-on projects, community engagement, and feedback prepare students to solve real coding problems, which is the end goal of dedicating time and money to these resources. Providers should adopt active learning and problem-solving skills to meet the demand for skilled programmers. This approach will help students gain the skills and confidence needed for success in software development.

References

Handur, V., Kotecha, K., Narayanan, K., Parekh, P., & Rathod, H. (2016). Integrating class and laboratory with hands-on programming: Its benefits and challenges. *2016 IEEE 4th International Conference on MOOCs, Innovation and Technology in Education (MITE)*, 163–168. https://doi.org/10.1109/MITE.2016.041

Kim, A. S., & Ko, A. J. (2017). A pedagogical analysis of online coding tutorials. In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education* (pp. 321–326). Association for Computing Machinery. https://doi.org/10.1145/3017680.3017728

Merrill, D. C., Reiser, B. J., Merrill, S. K., & Landes, S. (1995). Tutoring: Guided Learning by Doing. *Cognition and Instruction*, *13*(3), 315–372. https://doi.org/10.1207/s1532690xci1303_1

Anderson, J. R., Conrad, F. G., & Corbett, A. T. (1989). Skill acquisition and the LISP tutor. *Cognitive Science, 13*(4), 467–505. https://doi.org/10.1145/191033.191085

Yeom, S., Herbert, N., & Ryu, R. (2022). Project-based collaborative learning enhances students' programming performance. In *Proceedings of the 27th ACM Conference on Innovation and Technology in Computer Science Education Vol. 1* (pp. 248–254). Association for Computing Machinery. https://doi.org/10.1145/3502718.3524779

Bijker, W. E., Hughes, T. P., & Pinch, T. J. (Eds.). (1987). *The social construction of technological systems: New directions in the sociology and history of technology*. MIT Press.

Malik, K. M., & Zhu, M. (2023). Do project-based learning, hands-on activities, and flipped teaching enhance students' learning of introductory theoretical computing classes? *Education and Information Technologies, 28*, 3581–3604. https://doi.org/10.1007/s10639-022-11350-8

Reese, H. W. (2011). The learning-by-doing principle. *Behavioral Development Bulletin, 17*(1), 1–19. https://doi.org/10.1037/h0100597

Mayer, R. E. (1981). The psychology of how novices learn computer programming. *ACM Computing Surveys, 13*(1), 121–141. https://doi.org/10.1145/356835.356841

Xinogalos, S. (2016). Designing and deploying programming courses: Strategies, tools, difficulties, and pedagogy. *Education and Information Technologies, 21*, 559–588. https://doi.org/10.1007/s10639-014-9341-9

Van Merriënboer, J. J. G., & Krammer, H. P. M. (1987). Instructional strategies and tactics for the design of introductory computer programming courses in high school. *Instructional Science, 16*, 251–285. https://doi.org/10.1007/BF00120253