eISSN: 2229-3213 pISSN: 2229-3205
The Journal of Applied Science and Social Science
Volume 16,Issue 06 June, 2026

DEVELOPING STUDENTS' PROGRAMMING COMPETENCY THROUGH GAME CREATION IN JAVASCRIPT

Islomova Shaxnoza Isom qizi
I-year master's student
II- Bukhara State Pedagogical Institute
shaxnozaislomova3105@gmail.com
0009-0008-8770-7784, 998936868696
https://doi.org/10.5281/zenodo.20530174

Abstract. This thesis examines, from theoretical and methodological perspectives, the development of students' competencies in algorithmic thinking, problem-solving, and code writing through the creation of educational games in JavaScript. The principles of Game-Based Learning (GBL), the pedagogical potential of simple JavaScript games, and the ways to integrate them into the classroom are analyzed.

Keywords: JavaScript, game-based learning, gamification, programming competency, algorithmic thinking, constructivism, motivation, cognitive load.

Аннотация. В данной статье рассматривается теоретический и методологический аспект развития компетентности учащихся в области алгоритмического мышления, решения задач и написания кода посредством создания учебных игр на языке программирования JavaScript. Анализируются принципы обучения на основе игр (Game-Based Learning), педагогические возможности простых JavaScript-игр, а также способы их интеграции в учебный процесс.

Ключевые слова: JavaScript, обучение на основе игр, геймификация, компетентность в программировании, алгоритмическое мышление, конструктивизм, мотивация, когнитивная нагрузка.

In today's digital economy, programming skills have become a widely demanded professional competency. At the same time, traditional teaching practice shows that learning abstract syntax rules in isolation creates excessive cognitive load in students (Sweller, 1988) and reduces motivation. As a result, students perceive programming as a "difficult subject" and find no opportunity to apply it in practice. The Game-Based Learning (GBL) methodology offers a fundamental solution to this problem: students organically acquire programming concepts in a purposeful, interactive environment where results are visible in real time. JavaScript, with its ability to run directly in the browser, produce visual results, and its user-friendly syntax, is the most suitable tool for creating educational games. This thesis aims to substantiate which competencies are formed through JavaScript games and how this process can be pedagogically organized.

Theoretical Foundations of Game-Based Learning and Programming Competency

The pedagogy of learning through play is grounded in Vygotsky's (1978) concept of the Zone of Proximal Development (ZPD): optimal growth occurs when a student performs tasks that are slightly above their current knowledge boundary. The mechanism of progressively increasing difficulty in games automatically implements this ZPD principle. Papert's (1980) constructionism further develops this idea, proving that knowledge is more deeply assimilated in the process of creating a real artifact - that is, a working program.

Gee (2003) demonstrated that digital games contain 36 important learning principles. Among those particularly relevant for programming education are: instant feedback - the result is visible immediately after code is written; safe-to-fail - if the game does not work, there is no punishment, only a new attempt; contextualized problem - the syntax rule is learned in connection with the game's objective. Csikszentmihalyi's (1990) "flow state" theory substantiates that when students perform tasks appropriate to their abilities, maximum concentration and intrinsic motivation emerge - a state that naturally forms in a well-designed game.

Wing (2006), who analyzed programming competency structurally, divided it into four components under the term "computational thinking": decomposition, pattern recognition, abstraction, and algorithm design. Brennan and Resnick (2012) divided the practical dimension of competency into three levels: programming concepts (variables, loops, functions), practices (iterative testing, code reuse), and perspectives (algorithmic thinking, collaboration). All of these levels are simultaneously activated during the JavaScript game-creation process.

Three-Stage Model of Game Creation in JavaScript

It is recommended to organize game creation according to the "simple to complex" principle across three stages. Each stage opens a new cognitive layer for the student along with a new programming concept.

Stage 1 - Number Guessing Game:

This is the game with the most minimal structure, built with just 10–12 lines of code:

```
let sir = Math.floor(Math.random() * 100) + 1;
function taxmin() {
let n = Number(document.getElementById('son').value);
if (n < sir)      { xabar('Enter a larger number!'); }
else if (n > sir) { xabar('Enter a smaller number!'); }
else              { xabar('Correct! Well done!'); }
}
```

In these 8 lines, the student simultaneously learns: declaring variables and assigning values, generating random numbers via the Math object, writing functions, the if-else if-else structure, and accessing DOM elements. All concepts are not abstract - they are necessary for the game to work, which is why they are retained in memory.

Stage 2 - Keyboard-Controlled Object:

This stage introduces the canvas and event listeners concepts:

```
let x = 200, y = 280;
document.addEventListener('keydown', function(e) {
if (e.key === 'ArrowRight') x += 8;
if (e.key === 'ArrowLeft')  x -= 8;
if (e.key === 'ArrowUp')    y -= 8;
if (e.key === 'ArrowDown')  y += 8;
draw();
});
function draw() {
ctx.clearRect(0, 0, 400, 400);
ctx.fillStyle = '#3A86FF';
ctx.fillRect(x, y, 40, 40);
}
```

This code demonstrates in practice: coordinate systems, program state, event-driven programming, and the fundamentals of animation. The student sees the result immediately upon pressing a key - this instant feedback dramatically increases motivation.

Stage 3 - Physics and Collision Mechanics:

The third stage introduces object collision and physical simulation:

```
let tezlik = 0;
const tortish = 0.8;
function oyinTsikli() {
tezlik += tortish;
qahramon.y += tezlik;
if (qahramon.y + qahramon.h >= zamin) {
qahramon.y = zamin - qahramon.h;
tezlik = 0;
```

ekraniYangilash();
requestAnimationFrame(oyinTsikli);
}

At this stage, the student masters: the game loop, modeling physical changes, representing an object as a set of properties, and working with the requestAnimationFrame API. These concepts directly correspond to the core paradigms of modern programming.

The following table shows which competencies each stage game develops and which cognitive level of Bloom's Taxonomy it corresponds to:

Game Type	Competencies Developed	Cognitive Level (Bloom'
Number guessing	Variables, conditionals, functions, DOM	Understanding, Applicati
Object control	Events, coordinates, canvas, state	Analysis
Jumping game	Game loop, physics simulation, collision	Evaluation
Quiz game	Arrays, objects, timer	Evaluation, Creation
Free project	All concepts, API, OOP	Creation (Synthesis)

Table 1. Game types, competencies, and Bloom's Taxonomy levels

Advantages over the Traditional Method: A Comparative Analysis

The differences between the game-creation methodology and the traditional exercise format are defined by the following aspects. First, there is a difference in the source of motivation: in a traditional exercise, the student writes code to "complete the assignment"; in a game project, it is "so that my game works" - this is an intrinsic source of motivation that reinforces knowledge in the long term.

Second, the attitude toward errors differs fundamentally. In a traditional format, an error means losing points - it is a punishment. In a game environment, an error is immediately visible from the game not working, and the student searches for the cause themselves. This "safe-to-fail" environment organically develops debugging skills. Third, the context problem is resolved: the question "why do we need a for loop?" receives a concrete answer from the game - to display a crowd of enemies on the screen. Abstraction disappears; the concept is tied to a real purpose.

Furthermore, the game-creation process gives students a number of important skills beyond programming: project thinking (first plan, then implement), an iterative approach (every non-working version is information for the next), and consideration of the user's perspective (if friends play it and point out shortcomings - that is real user testing).

Proposed Pedagogical Model

Based on the research, the following step-by-step pedagogical model is proposed for developing competency through JavaScript games:

-**Result first, explanation second.** Show the student a ready, working game and begin with: "you will write this too." Motivation must come first.

-**Modifying code is easier than writing code.** Give a ready 10-line game. Let the student change the color, speed, and scores. This ensures maximum understanding with minimal writing.

-**Every week - one concept, one game element.** Once arrays are learned - add a list of prizes. Once functions are learned - add a new action. Each new concept is immediately incorporated into the game.

-**Error is a tool, not a punishment.** If the game stops working: ask "What do you think caused this?" Do not give the answer - show the way.

CONCLUSION

This thesis has scientifically substantiated the methodology of game creation in JavaScript by integrating Game-Based Learning theory with a competency-based approach.

The process of creating JavaScript games activates all cognitive levels of Bloom's Taxonomy - from understanding to creation - and develops programming competency in a contextualized environment. The three-stage model (number guessing - object control - physics game) progressively develops the student and allows them to master 6–8 new programming concepts in a practical context at each stage. The "safe-to-fail" principle in the game environment naturally fosters in students a culture of fearlessness toward mistakes, independent debugging, and experimental thinking.

REFERENCES

1. Brennan K., Resnick M. New Frameworks for Studying and Assessing the Development of Computational Thinking // Proc. AERA Annual Meeting. - Vancouver, 2012.
2. Deterding S., Dixon D., Khaled R., Nacke L. From Game Design Elements to Gamefulness: Defining «Gamification» // Proc. MindTrek 2011. - ACM, 2011. - P. 9–15.
3. Qian M., Clark K.R. Game-Based Learning and 21st Century Skills // Computers in Human Behavior. - 2016. - Vol. 63. - P. 50–58.
4. Sweller J. Cognitive Load During Problem Solving: Effects on Learning // Cognitive Science. - 1988. - Vol. 12(2). - P. 257–285.
5. Habibullayev I.I. Innovative Methods in Information Technology Education. - Tashkent: National Encyclopedia of Uzbekistan, 2019. - 198 p.
6. Nishonov M.N. Competency-Based Approach in Higher Education and Its Implementation Mechanisms. - Tashkent: TDPU Publishing House, 2021. - 152 p.
7. Ergashev O.T. Creating Educational Games in JavaScript and Their Educational Effectiveness // IT Journal of Uzbekistan. - 2022. - No. 4. - P. 18–26.
8. Xo'jayev B.R., Qodirov A.T. The Role of Digital Tools in Competency-Based Education // Pedagogy and Psychology. - 2023. - No. 2. - P. 44–52.
9. Decree of the President of the Republic of Uzbekistan on the "Digital Uzbekistan - 2030" Strategy. PF-6079. - Tashkent, 2020.
10. MDN Web Docs. JavaScript Guide. - URL: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide (accessed: 10.02.2025).
11. freeCodeCamp. JavaScript Algorithms and Data Structures. - URL: https://www.freecodecamp.org/learn/javascript-algorithms-and-data-structures