

# Algorithm Visualization Project

Interactive Data Structure and Algorithm Visualizations

A modern visualization toolkit for learning algorithms and data structures step by step.

Created by Ayoub Elouardi · Based on work by David Galles, University of San Francisco

# Meet the Team

 Maintainer: Ayoub Elouardi

 Original Creator: David Galles (University of San Francisco)

 Institution: ALX Software Engineering Program

## Project Goal

To deliver a compelling, educational, and interactive visualization resource for learners and educators.

# Project Overview



Comprehensive collection of interactive visualizations of data structures and algorithms.



Built with HTML5 Canvas and JavaScript.

## Mission

Make algorithms and data structures intuitive and accessible through animated and interactive demonstrations.

## Key Educational Benefits



Visualize abstract concepts



Step-by-step learning



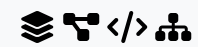
Ideal for classroom use



Self-guided exploration

# Technical Architecture Overview

---



Modular, object-oriented architecture.

Organized into Presentation, Algorithm, Animation, and Rendering layers.

Supports extensibility, maintenance, and high performance for browser-based execution.

# Technology Stack & Third-Party Tools



## HTML5 Canvas

2D rendering engine for all visual elements



## Vanilla JavaScript (ES5)

Core application logic with jQuery & jQuery UI for UI components



## CSS3

Styling and responsive layout design



## Browser-Only Solution

No backend required, runs entirely in the browser



## Open Source

All dependencies distributed under BSD 2-Clause License

# System Architecture Layers



## 1. Presentation Layer

HTML pages, CSS styling, UI controls (buttons, forms, interactive elements)



## 2. Algorithm Layer

Base Algorithm class, specific algorithm implementations (BST, AVL, etc.), command queue system



## 3. Animation Layer

AnimationMain.js engine, AnimatedObject classes (shapes/labels), ObjectManager (scene graph)



## 4. Rendering Layer

HTML5 Canvas (2D Context), DOM Events (user input), Timing API (animation frame control)

## Key Design Patterns



**Command Pattern:** Enables reproducible, undoable animations through atomic operations



**Scene Graph:** Hierarchical object organization for efficient animation orchestration

# Algorithm Implementation Patterns



## Base Algorithm Class

Provides initialization framework and command generation interface for all specific algorithms



## Linear Interpolation

Creates smooth transitions between states using calculated intermediate positions



## Command Queue

Stores atomic animation steps for playback, allowing granular control over visualization



## Undo/Redo System

Maintains action history for reversible operations during visualization



## Object-Oriented Modeling

Models animated elements as objects (circles, rectangles, labels, links) with inheritance



## Animation Control

Play, pause, step, and speed controls for precise visualization navigation

### Command Pattern Implementation Example:

```
// Adding a node to BST
this.cmd("CreateCircle", nodeID, value, x, y);
this.cmd("SetForegroundColor", nodeID, "#007700");
this.cmd("Move", nodeID, finalX, finalY);
this.cmd("Connect", parentID, nodeID, "#007700");
```

# Development Achievements & Successes



## **50+** algorithms and data structures visualized

Comprehensive collection spanning from basic to advanced computer science concepts



## **Real-time, user-controlled step-by-step animations**

Interactive playback with play, pause, and step controls for detailed learning



## **Platform-agnostic implementation**

Works seamlessly on desktop, mobile, and tablet devices



## **Extensible structure for new algorithms**

Modular design allows easy addition of new visualizations

## **Educational Impact**

Widely adopted in computer science classrooms and for self-study, making abstract algorithmic concepts tangible and accessible to students at all levels.



# Development Challenges



## Synchronizing animations and user controls

Ensuring smooth playback while maintaining step-by-step control



## Cross-browser compatibility

Creating fallbacks for older browsers while maintaining modern features



## Performance optimization for large data sets

Balancing visual fidelity with rendering speed for complex algorithms



## Robust input validation

Preventing errors while maintaining flexible input options for all algorithms



## Modularizing legacy code

Restructuring while maintaining stability and backward compatibility

### Core Challenge:

Balancing educational clarity with technical performance across diverse platforms and browsers

# Areas for Improvement & Lessons Learned



## Further modularization and adoption of ES6+

Refactoring to use modern JavaScript features: classes, modules, and async patterns



## Potential migration to TypeScript for type safety

Adding static typing to improve maintainability and catch errors early



## Improved mobile touch support

Enhancing gesture controls and responsive design for mobile users



## Enhanced documentation and user-guides

Creating comprehensive API docs and instructional content







## Community engagement for contributions

Building an open source community to expand algorithm coverage

## Key Lessons Learned

Balance between educational clarity and technical complexity is crucial when visualizing algorithms. Start with simpler implementations and iteratively refine based on user feedback.

# Next Steps & Future Development

-  **Advanced Algorithms**  
Add more advanced algorithm visualizations including neural networks, graph flows, and computational geometry algorithms
-  **Interactive Quizzes & Challenges**  
Integrate practice problems and challenges to test knowledge and understanding of algorithms
-  **Learning Analytics**  
Implement metrics to track user interactions and provide insights for educational assessment
-  **PWA Support**  
Develop Progressive Web App capabilities for offline access and improved mobile experience


## Development Roadmap

These enhancements aim to expand the project's educational impact while maintaining the core focus on algorithm visualization and interactive learning.

## Conclusion & Contact

The Algorithm Visualization Project makes abstract concepts tangible and engaging.

# Thank you for your attention!

 Contact: @ayoubelouardi (GitHub)

 Project Repository: [github.com/ayoubelouardi/algo-visual-project](https://github.com/ayoubelouardi/algo-visual-project)