Software Engineering

Winter Course 24-25

INTRODUCTION

This course is designed to teach students the foundational skills needed to build web pages and draw on them using HTML, the *Canvas* element, and JavaScript. The course will be divided into theoretical lessons, practical exercises, and hands-on projects. By the end, students will have the knowledge to build dynamic, interactive web pages and understand the principles of front-end development.

Prerequisites

This course is designed for beginners, but some familiarity with basic computer concepts and web browsing is helpful. The following are recommended prerequisites:

- **Basic Computer Literacy:** Comfort using a computer, installing software, and navigating web pages.
- **Familiarity with Text Editors:** Experience with simple text editors (e.g., Notepad or VSCode) for writing code.
- **Basic Knowledge of the Internet:** Understanding of how websites work, including using web browsers.
- **Optional:** Familiarity with basic programming concepts (e.g., variables, loops, and conditionals) is beneficial but not required.

Course Objectives

By the end of this course, students will be able to:

1. **Understand Web Development Fundamentals:**
    - Gain a solid understanding of how web pages work and the role of HTML, CSS, and JavaScript in web development.
2. **Build Web Pages with HTML:**
    - Create well-structured, accessible, and semantic HTML documents, embedding multimedia, links, forms, and more.
3. **Understand and Use the Canvas Element:**
    - Learn how to draw and manipulate graphics using the HTML <canvas> element and JavaScript.

4. **Write JavaScript for Interactive Web Pages:**
    - Understand core JavaScript concepts such as variables, functions, and control structures.
    - Use JavaScript to manipulate HTML elements and control the Canvas for drawing and animation.
5. **Integrate HTML, Canvas, and JavaScript:**
    - Build interactive and dynamic web applications that combine HTML structure, Canvas graphics, and JavaScript logic.
6. **Develop a Final Project:**
    - Apply all learned skills to develop a web-based drawing application or game, demonstrating competency in front-end web development.

Teaching Methodology

The course follows a project-based, hands-on approach that emphasizes practical learning through real-world examples and exercises. Here’s a breakdown of the teaching methodology:

- **Theory with Immediate Practice:**
    - **Lectures:** Short, targeted theoretical lessons introduce core concepts like HTML structure, the Canvas API, and JavaScript programming.
    - **Hands-on Coding:** After each theory lesson, students will practice by immediately applying what they’ve learned through guided exercises.
- **Incremental Learning:**
    - **Layered Learning Approach:** The course gradually introduces increasingly complex topics, starting with simple HTML pages, then adding Canvas drawing techniques, and finally integrating JavaScript to make web pages interactive.
    - **Building Blocks:** Each lesson builds upon previous ones, ensuring a solid **understanding before advancing to more challenging topics.**
- **Interactive Problem-Solving:**
    - **Live Coding Sessions:** Instructors will guide students through building code step-by-step, explaining each part as they go along.
    - **Code-Along:** Students will code along with instructors in real-time, helping them immediately apply what they learn.
- **Project-Based Learning:**
    - **Mini-Projects:** Throughout the course, students will complete small projects such as creating basic web pages, drawing shapes, or implementing user interactivity.
    - **Capstone Project:** The course culminates in a larger final project where students will create a fully functional web-based drawing application using all of the skills they've acquired.
- **Peer Learning and Collaboration:**

  - **Pair Programming:** Encourage students to work in pairs or small groups to solve coding challenges and develop projects together, fostering peer learning.
  - **Group Discussions:** Promote class-wide discussions and code reviews to share different approaches and solutions to common coding problems.
- **Real-World Application:**
  - **Case Studies:** The course will highlight real-world use cases of web applications built with HTML, Canvas, and JavaScript (e.g., interactive drawing tools, web-based games).
  - **Practical Scenarios:** Exercises are designed to reflect real-world challenges developers face when creating interactive web applications.
- **Assessment and Feedback:**
  - **Quizzes and Tests:** Regular quizzes will test students' understanding of theoretical concepts.
  - **Code Review:** The instructor will provide feedback on code structure, logic, and design, encouraging best practices in coding.
  - **Final Project Evaluation:** Students will present their final project for evaluation, receiving constructive feedback on their code and overall application design.

CONTENTS OF THE COURSE

Module 1: Introduction to Web Development & HTML (5 Hours)

Objective: Learn the basics of HTML and the structure of a web page.

Lesson 1: Overview of Web Development (1 Hour)

- What is web development?
- Overview of front-end vs back-end.
- Web development languages: HTML, CSS, JavaScript.
- How websites work (Client-Server model).

Lesson 2: Introduction to HTML (1 Hour)

- What is HTML and why it's used.
- The anatomy of an HTML document.
- Common tags and elements: `<html>`, `<head>`, `<title>`, `<body>`, `<h1>` to `<h6>`, `<p>`, `<a>`, `<img>`, `<ul>`, `<ol>`, `<li>`, etc.
- Practical Exercise: Build a basic webpage structure.

Lesson 3: HTML Document Structure & Semantic HTML (1 Hour)

- Understanding the Document Object Model (DOM).

- Semantic HTML and its importance: `<header>`, `<footer>`, `<section>`, `<article>`, `<nav>`, etc.
- Accessibility and SEO-friendly HTML.
- Practical Exercise: Create a simple blog layout using semantic HTML.

Lesson 4: Multimedia and Forms in HTML (1 Hours)

- Embedding multimedia: images, audio, video.
- HTML Forms: `<form>`, `<input>`, `<select>`, `<textarea>`, `<button>`.
- Form attributes, validation, and action.
- Practical Exercise: Create a contact form and embed a video on a webpage.

Lesson 5: CSS Overview (1 Hour)

- Introduction to CSS for styling HTML.
- Inline, internal, and external CSS.
- Basic CSS syntax (selectors, properties, and values).
- Practical Exercise: Apply basic CSS to an HTML webpage.

Module 2: Introduction to Canvas (5 Hours)

Objective: Learn how to use the `Canvas` element to draw graphics on web pages.

Lesson 6: Introduction to the Canvas Element (1 Hour)

- What is Canvas in HTML?
- Canvas vs SVG for drawing.
- The `<canvas>` element and its properties.
- Practical Exercise: Add a Canvas element to an HTML page.

Lesson 7: Working with the 2D Context (2 Hours)

- Accessing the 2D drawing context using JavaScript.
- Canvas coordinate system.
- Basic drawing functions: `fillRect()`, `strokeRect()`, `clearRect()`.
- Practical Exercise: Draw basic shapes (rectangles) on the canvas.

Lesson 8: Drawing Paths and Shapes (1 Hours)

- Drawing paths with `beginPath()`, `moveTo()`, `lineTo()`.
- Creating shapes: lines, rectangles, circles, and polygons.
- Filling and stroking shapes.
- Practical Exercise: Create a simple illustration using shapes.

Lesson 9: Colors, Gradients, and Patterns (1 Hours)

- Setting fill and stroke colors.
- Working with gradients: linear and radial.
- Applying patterns using images.

- Practical Exercise: Draw colorful and patterned backgrounds for a web page.

Module 3: Introduction to JavaScript (10 Hours)

Objective: Understand JavaScript fundamentals and how to manipulate the `Canvas` element using JavaScript.

Lesson 10: Introduction to JavaScript (2 Hours)

- What is JavaScript and its role in web development.
- Basic syntax: variables, data types, operators.
- Understanding the `<script>` tag and where to include it.
- Practical Exercise: Add simple JavaScript alerts and console messages to an HTML page.

Lesson 11: JavaScript Variables and Data Types (1 Hour)

- Declaring variables using `var`, `let`, `const`.
- Primitive data types: strings, numbers, booleans, arrays, and objects.
- Practical Exercise: Practice variable declarations and using different data types.

Lesson 12: JavaScript Functions and Control Structures (2 Hours)

- Defining and calling functions.
- Conditional statements: `if`, `else`, `switch`.
- Loops: `for`, `while`.
- Practical Exercise: Write functions to control logic flow in an application.

Lesson 13: Event Handling and DOM Manipulation (2 Hours)

- Handling user input: `onclick`, `onchange`, `onmouseover`.
- Accessing and modifying HTML elements using JavaScript (`document.getElementById()`, `querySelector`).
- Practical Exercise: Create interactive elements that change when clicked.

Lesson 14: JavaScript and the Canvas Element (3 Hours)

- Using JavaScript to control the canvas context.
- Drawing shapes dynamically using loops and events.
- Animation basics with `requestAnimationFrame()`.
- Practical Exercise: Create an interactive drawing application that reacts to user input (e.g., drawing on canvas using mouse events).

Module 4: Advanced Canvas and JavaScript Techniques (5 Hours)

Objective: Apply advanced JavaScript and Canvas techniques for more complex projects.

Lesson 15: Advanced Drawing Techniques (2 Hours)

- Drawing images and manipulating pixels.

- Image transformations: scaling, rotating, translating.
- Practical Exercise: Create a basic game-like canvas with moving objects.

Lesson 16: Animations and Games with Canvas (2 Hours)

- Understanding the game loop.
- Basic collision detection and user controls.
- Practical Exercise: Create a simple interactive game using the Canvas and JavaScript.

Lesson 17: Final Project: Building a Drawing Application (1 Hour)

- Objective: Combine everything learned to build a drawing app.
- Project:
  - Build an application where users can draw, choose colours, and save their drawings.
  - Present project.

Module 5: Capstone Project and Review (5 Hours)

Lesson 18: Capstone Project: Build an Interactive Web Page with Drawing Features (3 Hours)

- Project:
  - Create a complete web page with an interactive canvas where users can draw.
  - Incorporate HTML, Canvas, and JavaScript for full interactivity.

Lesson 19: Review and Assessment (2 Hours)

- Topics:
  - Review of HTML, Canvas, and JavaScript concepts.
  - Individual assessments and project presentations.

This course should provide a well-rounded introduction to modern front-end web development with a focus on HTML, Canvas, and JavaScript.

Attendance is obligatory as well as any required reading or assignments. Throughout the course, in lecture or discussion sections we expect respect of one another, a positive environment in this class is up to each of the participants. To that point, if computers/tablets or other devices are used during this time, they should be used solely for course material and note-taking, so as not to distract others.

At least 2 years of study in an Engineering degree completed before the program start.

SCHEDULE AND CREDITS

- Contact hours: 30 h.
- Estimated time for homework and study: 10 h.
- ECTS Credits: 3 (1,5 credits in USA).

LEARNING OUTCOMES

By the end of the course, students will be able to:

1. Build Structured and Accessible Web Pages Using HTML:

- Create well-organized HTML documents using semantic elements (e.g., headers, paragraphs, lists).
- Embed multimedia elements such as images, videos, and audio in web pages.
- Design and implement functional HTML forms for user input.

2. Understand and Implement the HTML Canvas Element for Drawing:

- Set up and configure the HTML `<canvas>` element within a web page.
- Use the 2D rendering context to draw basic shapes (rectangles, circles, lines) on the canvas.
- Apply styles, gradients, and patterns to Canvas drawings.

3. Use JavaScript to Create Interactive Web Pages:

- Write JavaScript code that includes variables, functions, loops, and conditionals.
- Manipulate HTML elements using JavaScript (e.g., changing content or styles dynamically).
- Respond to user interactions (e.g., mouse clicks, keyboard input) through event handling.

4. Draw and Animate Graphics Using JavaScript with Canvas:

- Dynamically draw shapes, paths, and text on the Canvas using JavaScript.
- Implement basic animations using the `requestAnimationFrame()` method.
- Create interactive drawings that respond to user input such as mouse movements or clicks.

5. Develop Interactive and Visual Web Applications:

- Combine HTML, Canvas, and JavaScript to build a fully interactive web application (e.g., a drawing tool or simple game).
- Implement user-friendly features like color selection, undo, and save functionality for drawings.

6. Demonstrate Problem-Solving and Debugging Skills:

- Use debugging tools in the browser to identify and fix issues in HTML and JavaScript code.
- Break down complex tasks into manageable coding steps and solve problems methodically.

7. Apply Best Practices in Front-End Web Development:

- Write clean, readable, and well-commented HTML and JavaScript code.
- Follow web standards and ensure accessibility through semantic HTML.
- Understand how to structure projects and manage external assets like images and scripts.

8. Complete a Capstone Project:

- Design and build a full-featured web-based drawing or animation application as a final project.
- Present and demonstrate the application, showcasing the integration of HTML, Canvas, and JavaScript.

Key Competencies:

- Proficiency in HTML for web page structure.
- Skill in using the Canvas element for drawing graphics.
- Fundamental JavaScript programming and DOM manipulation.
- Experience building real-world interactive web applications.

EVALUATION

To ensure that students gain a comprehensive understanding of the course material and can apply their knowledge effectively, the course evaluation will be based on a combination of practical projects, assessments, and participation. Here’s a breakdown of the evaluation and grading criteria:

1. Class Participation and Engagement (10%)

Objective: Encourage active involvement during the course through discussions, coding exercises, and peer collaboration.

Criteria:

- Attendance and punctuality in class sessions.
- Active participation in live coding sessions, group discussions, and Q&A.
- Willingness to engage in pair programming and provide feedback to peers.

2. Weekly Quizzes and Assignments (20%)

Objective: Test theoretical understanding and practical application of HTML, Canvas, and JavaScript concepts covered in weekly lessons.

Criteria:

- Weekly quizzes focused on key concepts (HTML structure, Canvas basics, JavaScript functions).

- Coding assignments that require students to build simple web pages, implement Canvas drawings, and manipulate DOM elements.
- Timely submission of assignments.

3. Mini-Projects (30%)

Objective: Allow students to apply their learning in real-world scenarios through project-based learning.

Criteria:

- Project 1 (HTML & Semantic Structure): Create a multi-page static website with proper use of HTML tags, semantic elements, and media embedding. (10%)
- Project 2 (Canvas Drawing): Build a simple drawing application using Canvas and JavaScript, where users can draw basic shapes (rectangles, circles). (10%)
- Project 3 (JavaScript Interaction): Create an interactive web page where JavaScript handles user input (clicks, mouse movement) and updates the content or Canvas drawing dynamically. (10%)

Projects will be evaluated on code quality, creativity, functionality, and adherence to best practices.

4. Final Capstone Project (30%)

Objective: Showcase students' ability to integrate all the course concepts into a single, comprehensive project.

Criteria:

- **Project:** Build a full-featured web-based drawing or animation application using HTML, Canvas, and JavaScript.
- The application should include:
  - Proper HTML structure and user interface (UI).
  - Use of Canvas to create interactive drawings or animations.
  - JavaScript logic for user input (e.g., drawing with a mouse, selecting colors, undo/redo functionality).
  - Presentation of the project, explaining how the concepts were integrated.
- **Grading:** The project will be assessed based on:
  - Functionality (15%): How well the application works (e.g., smooth drawing, responsive interface).
  - Code Quality (10%): Clean, well-organized, and well-commented code.
  - User Experience (5%): Intuitive interface design, ease of use, and overall user-friendliness.

5. Final Written or Practical Exam (10%)

Objective: Assess overall understanding of course concepts through a formal evaluation.

Criteria:

- - A combination of multiple-choice, short answer, and coding exercises.
- - Topics will include HTML semantics, Canvas API usage, and JavaScript logic.

GRADING SCALE

This evaluation framework balances theory and practice, rewarding engagement, consistent effort, and the ability to apply knowledge to practical projects.

Final Grade Composition

- Class Participation: 10%
- Quizzes and Assignments: 20%
- Mini-Projects: 30%
- Final Capstone Project: 30%
- Final Exam: 10%

Scale

Grade	*Percentage*
A+	90-100 %
A	85-89%
B+	80-84%
B	75-79%
C+	70-74%
C	65-69%
D	60-64%
F	< 60%