
Frontend Web Development
</> for </>
Youngsters
Suggested for Age 12-17
FRONTEND WEB DEVELOPMENT
EMOSTEL ACADEMY

FRONT-END WEB DEVELOPMENT FOR YOUNGSTERS 12+

A comprehensive curriculum

COURSE OVERVIEW

Perfect for young tech enthusiasts aged 10-16, this course introduces them to the exciting world of web development. They will learn how to use HTML to structure web content, CSS to style it beautifully, and JavaScript to make it interactive and fun. Through engaging, hands-on projects, they'll create their very own web pages and see their ideas come to life on the internet.

Emostel Academy

Break Into Tech: iCode For Children
https://emostelacademy.org/bit/icode
+2348141897754, +2349077471007
Course Tutor: Mr. Emmanuel Kadiri

Course Details

Welcome to the **"Young Coders: Frontend Web Development"** course! This engaging and interactive program is designed specifically for kids to explore the exciting world of web development. Through a blend of live sessions, pre-recorded videos, and hands-on projects, students will learn how to build stunning and functional websites from scratch. The course fosters creativity, logical thinking, and problem-solving skills, preparing young learners for the digital age while keeping the learning process fun and inspiring.

COURSE OBJECTIVES:

This course is designed to fulfill the following objectives:

1. **Introduce Students to the Fundamentals of Frontend Web Development:** Provide an overview of key concepts and tools used in creating websites.

2. **Develop Proficiency in HTML, CSS, and JavaScript:** Equip students with the skills to build and style websites using essential web development languages.

3. **Encourage Creative Thinking Through Designing and Building Interactive Web Pages:** Inspire students to express their creativity by designing and developing engaging and interactive websites.

4. **Teach Best Practices in Web Design, Usability, and Accessibility:** Instil the importance of creating websites that are user-friendly, visually appealing, and accessible to all users.

5. **Build a Foundational Understanding That Can Be Expanded Upon in Future Advanced Courses:** Lay the groundwork for further learning in web development and related fields.

6. **Foster Collaborative Skills Through Group Projects and Community Engagement:** Promote teamwork and communication by encouraging collaboration on projects and active participation in the learning community.

7. **Develop Analytical Skills:** Through our Math Clinic sessions, students will enhance their logical thinking and problem-solving abilities, which are crucial in programming and web development.

8. **Cultivate Leadership Qualities:** Our Leadership Training sessions aim to build confidence, teamwork, and communication skills, preparing students to lead group projects and collaborate effectively in tech environments.

9. **Integrate Cross-Disciplinary Learning:** Encourage students to apply mathematical concepts in coding challenges and leverage leadership skills in project management and presentations.

10. **Holistic Development:** Foster a well-rounded educational experience by combining technical skills with essential life skills, preparing students for future academic and personal success.

LEARNING OUTCOMES

1. **Understand and Apply Web Development Fundamentals:** Proficiently use HTML, CSS, and JavaScript to build and style websites.
    - **Understand and Apply the Basic Concepts of HTML to Structure Web Content:** Learn to create and organize content on a webpage using HTML tags.
    - **Use CSS to Style and Layout Web Pages Effectively:** Apply styling techniques to enhance the appearance and layout of web pages.
    - **Implement Interactive Features Using JavaScript:** Add dynamic and interactive elements to web pages through JavaScript coding.

2. **Build Responsive Websites that Function Across Various Devices and Screen Sizes:** Develop websites that adapt seamlessly to different devices and screen resolutions.

3. **Utilize Development Tools and Environments Efficiently:** Gain familiarity with essential web development tools and coding environments to streamline the development process.

4. **Debug and Troubleshoot Common Web Development Issues:** Identify and fix common coding errors to ensure smooth website functionality.

5. **Develop a Personal Portfolio Website Showcasing Their Projects:** Create a personalized website that highlights and organizes completed projects and skills.

6. **Collaborate with Peers on Web Development Projects:** Work effectively in teams, sharing tasks and responsibilities to complete web projects collaboratively.

7. **Understand the Principles of User Experience (UX) and User Interface (UI) Design:** Learn the fundamentals of designing user-friendly and visually appealing websites.

8. **Prepare for More Advanced Studies in Web Development and Related Fields:** Build a strong foundation to pursue further education and careers in web development and technology

TARGET AUDIENCE:

- **Age Group:** 12 to 17 years Old
- **Prerequisites:** No prior coding experience required. A basic understanding of computer operations and internet usage is beneficial.

COURSE DURATION:

- **Total Duration:** 24 weeks (6 months)
- **Live Sessions:** Twice a week (1 hour per session)
- **Pre-recorded Videos:** Released weekly (2-3 hours of content per week)
- **Study Hours:** Recommended 4-5 hours per week (including live sessions and self-study)

SUPPORT AND COMMUNITY:

1. **Live Sessions:** Interactive classes twice a week where students can ask questions and engage directly with instructors.

2. **Pre-recorded Videos:** Comprehensive weekly lessons that students can watch at their own pace.

3. **Community Forum:** An online platform where students can interact, share ideas, collaborate on projects, and help each other solve problems.

4. **Tutor Support:**
    - **Office Hours:** Dedicated time slots where tutors are available for one-on-one assistance.
    - **Email Support:** Students can reach out to instructors for help and clarification.
    - **Feedback:** Regular constructive feedback on assignments and projects to aid improvement.
5. **Parental Updates:** Periodic reports on student progress to keep parents informed and engaged.

EVALUATION METHODS:

- **Assignments:** Weekly tasks that reinforce the lessons learned and encourage practice.
- **Quizzes:** Short quizzes at the end of each module to assess understanding of key concepts.
- **Projects:** Three major in-course projects, culminating in a final comprehensive project that showcases the student's skills. 10 after-course projects available for further mastery and demonstration of acquired skills.

CERTIFICATION:

Participants who successfully complete the course will be awarded a certificate of achievement. This certificate formally recognizes the skills and knowledge acquired. To earn this certificate, participants must complete all course modules and tasks, achieve the passing mark on quizzes, and submit an end-of-course project that passes assessment and approval.

COURSE FEE:

~~₦~~100,000 (one-off payment) or ~~₦~~40,000 (paid monthly for the first three months only)

ENROLMENT INFORMATION:

- **Registration Deadline:** Application is open from now till September 1, 2024
- **Course Start Date:** September 2, 2024

PROGRAM FEATURES:

- **Hands-On Learning:** We focus on interactive sessions where students design in real-time.
- **Project-Based Approach:** We reinforce concepts with small projects after each lesson and a final project at the end.
- **Gamification:** We introduce design challenges, quizzes, and fun competitions to keep students engaged.
- **Visual Aids and Examples:** We use diagrams, flowcharts, and live coding demonstrations to explain complex topics.
- **Peer Learning:** We encourage students to work together on exercises and projects to foster collaboration.
- **Regular Assessments:** We conduct short quizzes and design exercises to assess understanding and provide feedback.
- **Community Support:** Learners are added to communities pertaining to their course, so that they can interact with other learners to work together on projects, share ideas, express their challenges to get resolution, and get support from our expert tutors and community managers.
- **Direct Tutor Support:** Learners can also get support directly from the course tutor in cases where the tutor's attention is specially needed or on matters that may not be community resolvable.
- **Mathematics Clinic:** We clearly understand how math relates to coding and how it greatly impacts the Intelligence Quotient (IQ) of students and therefore deemed it necessary to introduce it into our programs. We also offer to (ethically) aid students with their homework challenges especially in math.

- **Soft Skills:** As an additional extracurricular activity, we prioritize training students in essential soft skills, with a strong focus on leadership. This extends to other crucial abilities such as teamwork, collaboration, communication, etc. Not only does this enrich their learning experience, but it also helps them stand out in exceptional and admirable ways, preparing them for remarkable future success.

- **Guest Speakers:** Periodic sessions with industry professionals to inspire and provide real-world insights.

- **Flexible Learning:** Content is accessible anytime, allowing students to learn at their own pace alongside structured live sessions.

- **Future Pathways:** Guidance on continuing education in technology fields and exploration of potential career paths in web development and design.

- **Parental Engagement:** Opportunities for parents to participate in certain sessions and view student progress, fostering a supportive learning environment at home.

TECHNICAL REQUIREMENTS

Device Requirements:

Computer/Laptop:

- **Operating System:** Windows 10 or higher, macOS 10.13 (High Sierra) or higher, or any recent Linux distribution.
- **Processor:** Intel Core i3 or equivalent.
- **Memory (RAM):** 4 GB minimum (8 GB recommended).
- **Storage:** At least 10 GB of free disk space.
- **Display:** 13-inch screen with a resolution of 1366 x 768 pixels or higher.
- **Internet Connectivity:** Reliable broadband connection for downloading resources and attending live sessions.

Optional Hardware:

- **External Mouse:** Recommended for easier navigation and coding, especially for younger students.

- **Webcam and Headset with Microphone:** Useful for participating in live sessions and discussions.

Software Requirements:

- **Code Editor:** VS Code (free download) or any other preferred text editor.
- **Web Browsers:** Latest versions of Google Chrome, Mozilla Firefox, or Microsoft Edge.

Online Platforms and Other Tools:

1. **Learning Management System (LMS):**
   - The course will be hosted on the Emostel Academy's LMS platform. Students will need a registered account to access course content, assignments, and live sessions.

2. Video Conferencing Software:

- **Zoom/Google Meet:** For live classes, discussions, and Q&A sessions. Students should have the software installed and be familiar with joining meetings.

3. Collaboration Tool:

- **Slack, Discord, or Google Workspace:** Enable group work, brainstorming, and project collaboration.

4. Design Tools:

- Canva or Figma for creating design mockups (free versions suffice).

5. Version Control:

- GitHub account for project hosting and version control (tutorials will be provided).

6. Miscellaneous:

- Notebook and pen for taking notes during sessions.
- An eagerness to learn and explore!

CONTACT INFORMATION:

For more information or any queries, please contact us at:

- **Email:** study@emostelacademy.org
- **Phone:** +234-814-189-7754, +234-907-747-1007
- **Website:** https://emostelacademy.org/

Emostel Academy is a brand of **Emostel Skills and Development Limited** RC: 1632636

Module 1:

Introduction to Web Development

Lesson 1:

- Introduction to the Internet and the World Wide Web
- Understanding how websites work
- Overview of frontend vs. backend development
- Setting up the development environment

TASKS:

1. Research and write a short summary of how websites work and the difference between frontend and backend development.
2. Set up your development environment and share a screenshot of your workspace.
3. Create a basic HTML document with a "Hello World" message and submit the code.

Lesson 2:

- Introduction to HTML
- Basic HTML tags and structure
- Creating your first simple webpage

TASKS:

1. Create a simple webpage with a title, headings, and a paragraph introducing yourself.
2. Find an image online and add it to your webpage using the <img> tag.
3. Add a hyperlink to your favourite website in your webpage and submit the HTML file.

Module 2:

HTML Fundamentals

Lesson 3:

- Structuring content with headings, paragraphs, and lists
- Adding images and links

TASKS:

1. Create a webpage that includes a list of your favourite hobbies using ordered and unordered lists.
2. Add images to each list item to visually represent your hobbies.
3. Create a navigation menu with links to different sections of your webpage.

Lesson 4:

- Tables and forms in HTML
- Semantic HTML and its importance

TASKS:

1. Design a simple contact form with input fields for name, email, and message.
2. Add a table to your webpage that displays a weekly schedule of your activities.
3. Refactor your HTML code to use semantic elements like <header>, <nav>, <article>, and <footer>.

Lesson 5:

- Multimedia elements: audio and video
- Project: Creating a personal profile page using HTML

TASKS:

1. Embed a video and an audio file into your webpage.
2. Create a personal profile page that includes an image, a short bio, and links to your social media profiles.
3. Submit your HTML project for review, ensuring all elements are correctly structured.

Module 3:

CSS Styling and Layouts

Lesson 6:

- Introduction to CSS
- CSS syntax and selectors
- Applying styles to HTML elements

TASKS:

1. Write a CSS stylesheet to style your personal profile page, changing colours, fonts, and background.
2. Use CSS selectors to apply different styles to headings, paragraphs, and links.
3. Add hover effects to your navigation menu using CSS.

Lesson 7:

- Colours, backgrounds, and fonts
- Box model concept

TASKS:

1. Apply padding, margins, and borders to elements on your page to improve layout.
2. Create a custom button style using CSS and apply it to your contact form.
3. Implement the box model by adding a styled border and background to one section of your page.

Lesson 8:

- Layouts with CSS: display, position, float
- Flexbox basics

TASKS:

1. Design a multi-column layout using Flexbox for your profile page's main content and sidebar.
2. Use CSS to style a navigation bar and make it horizontal.
3. Create a responsive grid layout for a gallery section on your page.

Lesson 9:

- Responsive design principles
- Media queries
- Project: Styling the personal profile page with CSS

TASKS:

1. Write media queries to make your webpage responsive for mobile devices.
2. Test your webpage on different screen sizes and adjust styles as needed.
3. Submit your fully styled and responsive personal profile page.

Module 4:

Advanced CSS and Design Principles

Lesson 10:

- CSS Grid layout
- Advanced selectors and combinators

TASKS:

1. Create a CSS Grid layout for a simple webpage with a header, main content area, and footer.
2. Use advanced CSS selectors like **:nth-child** and **:hover** to add special styles.
3. Apply a hover transition effect to buttons or images on your webpage

Lesson 11:

- CSS animations and transitions
- Using icon libraries and fonts

TASKS:

1. Create an animation that moves or changes the colour of an element on your webpage.
2. Incorporate an icon library like Font Awesome to add icons to your navigation menu or buttons.
3. Redesign your profile page following basic UX/UI principles, focusing on user-friendliness.

Lesson 12:

- Introduction to UI/UX design principles
- Designing for accessibility
- Project: Creating a responsive and interactive webpage

TASKS:

1. Ensure your webpage meets accessibility standards by adding alt text to images and ensuring proper contrast.
2. Review your webpage's design for consistency and usability, making any necessary adjustments.
3. Submit your final project for peer review, focusing on design principles and accessibility.

Module 5:

Introduction to JavaScript

Lesson 13:

- Understanding programming concepts
- Introduction to JavaScript syntax

TASKS:

1. Write a simple JavaScript program that outputs your name and age in the console.
2. Create a webpage with a button that, when clicked, displays a greeting message.
3. Use JavaScript to perform basic arithmetic operations and display the result on your webpage.

Lesson 14:

- Functions and control structures (if statements, loops)
- Events and event handling

TASKS:

1. Write a JavaScript function that takes two numbers as input and returns their sum.
2. Create an interactive webpage where clicking a button changes the color of a background element.
3. Use JavaScript to create a simple quiz with multiple-choice questions and display the score.

Lesson 15:

- Working with the DOM (Document Object Model)
- Manipulating HTML and CSS with JavaScript

TASKS:

1. Manipulate the DOM to change the content of an HTML element dynamically.
2. Create a to-do list application where users can add and remove items.
3. Use JavaScript to validate a form, ensuring all required fields are filled out before submission.

Lesson 16:

- Form validation using JavaScript
- Project: Adding interactivity to previous projects with JavaScript

TASKS:

1. Implement a JavaScript function to calculate the total price in a shopping cart based on user input.
2. Use JavaScript to show and hide content on your webpage based on user actions.
3. Submit your interactive webpage project, ensuring it includes at least one form, button, and dynamic content.

Module 6:

Advanced JavaScript Concepts

Lesson 17:

- Arrays and objects
- Introduction to JSON

TASKS:

1. Create an array and write a JavaScript function to sort its elements in alphabetical order.
2. Build a simple web application that stores user data in objects and displays it dynamically.
3. Explore JSON and write a script to parse and display JSON data on a webpage.

Lesson 18:

- Fetching data from APIs
- Asynchronous programming basics

TASKS:

1. Fetch data from a public API and display the information on your webpage (e.g., weather, movies).
2. Write a JavaScript function that asynchronously loads additional content when a button is clicked.
3. Create a basic single-page application (SPA) that loads content dynamically without reloading the page.

Lesson 19:

- Introduction to JavaScript libraries (e.g., jQuery)
- Project: Building a simple web application that consumes an API

TASKS:

1. Integrate a JavaScript library like jQuery into your project and use it to add animations or effects.
2. Write a script that dynamically generates content based on user input (e.g., creating new HTML elements).
3. Submit a small web application that incorporates an external API and provides a useful function.

Module 7:

Building Projects and Portfolio

Lesson 20:

- Planning and designing a complete website
- Wireframing and mockups

TASK:

1. Brainstorm and outline your final project idea, including the purpose, target audience, and main features.
2. Create wireframes or mockups for your final project, detailing the layout and design.
3. Begin setting up the HTML structure of your project, focusing on content organization.

Lesson 21:

- Developing the frontend of the planned website
- Implementing advanced features and interactivity

TASK:

1. Develop the main content sections of your project, incorporating HTML and CSS.
2. Start adding interactive features using JavaScript, such as forms, buttons, and dynamic content.
3. Test your project across different devices and browsers, noting any issues.

Lesson 22:

- Testing and debugging the website
- Ensuring cross-browser compatibility and responsiveness

TASK:

1. Refine your project by debugging and optimizing code for performance and usability.
2. Implement any remaining features and ensure that your project meets all design and functionality goals.
3. Prepare a short presentation of your project, explaining your design choices and technical implementation.

Lesson 23:

- Deploying the website to the internet
- Introduction to hosting services and version control (basic Git concepts)

TASK:

1. Deploy your project to a hosting platform and share the live link.
2. Review peer projects and provide constructive feedback on design, functionality, and user experience.
3. Finalize and submit your complete project for evaluation.

Module 8:

Course Wrap-up and Showcase

Lesson 24:

- Final project presentations
- Peer reviews and feedback sessions
- Discussing next steps and further learning resources
- Awarding certificates of completion

TASKS:

1. Present your final project to the class, showcasing your work and explaining your development process.
2. Participate in peer reviews, offering feedback and suggestions to classmates.
3. Reflect on your learning journey and set goals for future development and learning.

10-END-OF-COURSE PROJECTS

End-of-course projects for a Frontend Web Development course for kids should be fun, creative, and challenging enough to showcase their learning while being achievable within their skill level. Here are some project ideas we carefully gathered to help students apply the concepts they've learned throughout the course:

1. Personal Portfolio Website

- **Description:** Students create their personal portfolio website to showcase their projects and skills.
- **Key Features:**
  - Home page with an introduction and a short bio.
  - Project showcase section with descriptions, images, and links to live projects.
  - Contact form that uses JavaScript for validation.
  - Responsive design for viewing on different devices.
- **Skills Applied:** HTML, CSS, JavaScript, responsive design, form validation.

2. Interactive Quiz Game

- **Description:** Build an interactive quiz game where users answer multiple-choice questions on a topic of their choice.
- **Key Features:**
  - Dynamic question loading and randomization.
  - Score tracking and feedback after each question.
  - Timer for added challenge.
  - Use of JavaScript to handle user interactions and logic.
- **Skills Applied:** HTML, CSS, JavaScript, event handling, DOM manipulation.

3. Blog Website

- **Description:** Create a simple blog where students can post articles on topics they are passionate about.
- **Key Features:**
  - Home page with a list of blog posts.
  - Individual pages for each blog post with comments section.
  - Admin section (basic) to add new blog posts (simulated with local storage or static content).
  - Responsive design and use of CSS Grid or Flexbox for layout.
- **Skills Applied:** HTML, CSS, JavaScript, local storage, responsive design.

4. Online Storefront

- **Description:** Develop a basic online store interface where users can browse products, add items to a cart, and simulate a checkout process.
- **Key Features:**
  - Product listing page with images, descriptions, and prices.
  - Shopping cart functionality with item addition/removal.
  - Simple checkout form with JavaScript validation.
  - Responsive design for mobile and desktop users.
- **Skills Applied:** HTML, CSS, JavaScript, event handling, DOM manipulation, responsive design.

5. Weather App

- **Description:** Create a weather application that allows users to input a city and receive real-time weather updates.
- **Key Features:**
  - Input field for users to enter the city name.
  - Integration with a weather API to fetch current weather data.
  - Display of temperature, weather conditions, and an icon representing the weather.
  - Option to display the weather for the user's current location.
- **Skills Applied:** HTML, CSS, JavaScript, APIs, asynchronous programming.

6. Digital Art Gallery

- **Description:** Build an online gallery to display digital artwork created by the student or sourced from free online resources.
- **Key Features:**
  - Grid layout to display artwork thumbnails.
  - Lightbox effect to view artwork in full size.
  - Filter options (e.g., by category or artist).
  - Responsive design and use of advanced CSS for layout and effects.
- **Skills Applied:** HTML, CSS, JavaScript, responsive design, advanced CSS techniques.

7. Simple Web-Based Calculator

- **Description:** Design and develop a simple, functional calculator that can handle basic arithmetic operations.
- **Key Features:**
  - User-friendly interface with buttons for numbers and operations.
  - Real-time calculation display.
  - Clear function to reset calculations.
  - Use of JavaScript for handling arithmetic logic.
- **Skills Applied:** HTML, CSS, JavaScript, event handling, DOM manipulation.

8. Recipe Website

- **Description:** Create a website where users can browse, search, and view recipes.
- **Key Features:**
  - Home page with featured recipes.
  - Individual recipe pages with ingredients, instructions, and images.
  - Search functionality to filter recipes by name or ingredient.
  - Responsive design for a seamless experience across devices.
- **Skills Applied:** HTML, CSS, JavaScript, responsive design, form handling.

9. Customizable Greeting Card

- **Description:** Develop a web application where users can create and customize their digital greeting cards.
- **Key Features:**
  - Selection of card templates and designs.
  - Input fields for personalized messages.
  - Option to change colors, fonts, and backgrounds.
  - Downloadable or shareable final card.
- **Skills Applied:** HTML, CSS, JavaScript, DOM manipulation, responsive design.

10. Simple To-Do List Application

- **Description:** Create a web-based to-do list app where users can add, edit, and delete tasks.
- **Key Features:**
  - Task creation with an input field.
  - Option to mark tasks as completed.
  - Dynamic updating of task list using JavaScript.
  - Save tasks to local storage so they persist after page reloads.
- **Skills Applied:** HTML, CSS, JavaScript, local storage, DOM manipulation.

These projects not only reinforce the concepts taught in the course but also allow students to express their creativity and problem-solving skills. They can choose one or more project that aligns with their interests, making the learning process more enjoyable and personalized.

PLEASE NOTE:

Please note the following:

1. *As deemed necessary, the curriculum outlined above may undergo further modifications, including additions or deletions. Regardless of the situation, these changes are intended to better fulfil the objectives of the program.*
2. *The extra-curricular activities/sessions (e.g. leadership, soft skills, mathematics clinic sessions, etc.) are embedded across the learning path. Every session will be announced ahead.*
3. *These projects not only help students apply what they've learned but also allow them to express their creativity and build a portfolio of work that they can be proud of. We shall also provide the necessary project guidelines to each student and as well monitor their progress and help them with their challenges in appropriate and essential ways.*

THANK YOU FOR CHOOSING TO LEARN/TRAIN WITH US. WE ASSURE YOU OF AN AMAZING AND UNFORGETTABLE EXPERIENCE.

SEE YOU IN CLASS