

TEACHER EDITION • 2026 CURRICULUM

Web Development Mastery: From **Frontend** to Backend

A complete 4-module lesson plan covering HTML, CSS, JavaScript and Introduction to Backend — designed for teachers to deliver and students to build real projects.

FOR TEACHERS

Includes lesson objectives, code examples, classroom activities, homework & assessment rubrics. No prior teaching experience needed.

FOR STUDENTS

Beginner-friendly. By the end, students will have built a personal portfolio website and connected it to a live backend.

4 Modules

8-12 SESSIONS

15+ Projects

HANDS-ON

Beginner

ZERO TO HERO

Code Included

COPY & RUN

Course Overview

Goal: Equip students with practical skills to build modern, responsive websites and understand how the web really works behind the scenes. This pack is designed so any teacher can facilitate even without deep coding background — all explanations are in plain English.

Learning Outcomes

- ✓ Build semantic HTML pages from scratch
- ✓ Style professional layouts with CSS & Flexbox
- ✓ Add interactivity with JavaScript & DOM
- ✓ Explain client vs server, APIs, databases
- ✓ Deploy a full-stack mini project

Teaching Format

Audience: Secondary / High School / Bootcamp Beginners (Age 13+)

Duration: 4 Weeks (2 sessions per week, 90 min each)

Tools Needed: Laptop, VS Code (free), Browser (Chrome), Internet

Method: 20% Theory, 80% Live Coding + Practice

TABLE OF CONTENTS

Module 1: HTML — The Skeleton of the Web	Page 03
Module 2: CSS — The Styling & Design	Page 05
Module 3: JavaScript — The Brain & Behavior	Page 07
Module 4: Introduction to Backend	Page 09
Capstone Project: Portfolio + API	Page 11
Assessment Rubrics & Teacher Notes	Page 12

Module 1: HTML — The Skeleton

Duration: 2 Sessions (3 hours) | **Objective:** Students understand that HTML provides structure, not design.

1.1 WHAT IS HTML?

HTML (HyperText Markup Language) is like the skeleton of a human body. It defines WHAT is on the page: headings, paragraphs, images, links, buttons. Browsers read HTML and display content.

Teacher Analogy 🏠

Tell students: "Think of a house. HTML = bricks, walls, doors (structure). CSS = paint, curtains (design). JavaScript = electricity & doorbell (function)."

Key Terms

- **Tag:** `<p> </p>`
- **Element:** Tag + Content
- **Attribute:** Extra info: ``

1.2 BASIC STRUCTURE

HTML - INDEX.HTML

```
<!DOCTYPE html>
<html>
  <head>
    <title>My First Website</title>
  </head>
  <body>
    <h1>Hello Skoolyn!</h1>
    <p>This is my first webpage.</p>
  </body>
</html>
```

1.3 ESSENTIAL TAGS STUDENTS MUST KNOW

TAG	PURPOSE	EXAMPLE
<h1> to <h6>	Headings (h1 biggest)	Welcome
<p>	Paragraph	Text here
<a>	Link	Google
	Image	
, ,	Lists	Item
<div>	Container/Section	Groups content
<form>, <input>, <button>	Forms	Login, Contact forms
<header>, <nav>, <footer>	Semantic layout	Modern SEO-friendly tags

📌 CLASS ACTIVITY (25 MIN)

Task: Students create about.html with: Their name as h1, short bio as p, 3 hobbies as ul list, photo (img), link to their favorite website.

Teacher Tip: Let them open file directly in browser. No server needed yet. Celebrate every small win!

! **Common Mistake:** Students forget closing tags. Teach them VS Code auto-closes tags. Use Emmet: type ! + Tab to generate boilerplate.

Module 2: CSS — The Styling

Duration: 2 Sessions | **Objective:** Turn ugly HTML into a beautiful, responsive design.

2.1 WHAT IS CSS?

CSS = Cascading Style Sheets. Controls colors, fonts, spacing, layout, animations. If HTML is the skeleton, CSS is the clothing and makeup.

2.2 THREE WAYS TO ADD CSS

METHOD	HOW	BEST FOR
Inline	<code><p style="color:red;"></code>	Quick test (not recommended)
Internal	<code><style></code> in <code><head></code>	Single page demos
External (Best)	<code><link rel="stylesheet" href="style.css"></code>	Real projects - Always use this

CSS - style.css

```
body {
  font-family: Arial, sans-serif;
  background: #f8fafc;
  color: #1e293b;
}

.card {
  background: white;
  padding: 20px;
  border-radius: 12px;
  box-shadow: 0 4px 12px rgba(0,0,0,0.08);
  margin: 20px 0;
}

h1 {
  color: #0ea5e9;
  text-align: center;
}
```

2.3 THE BOX MODEL (MOST IMPORTANT CONCEPT)

Every element is a box with: Content → Padding → Border → Margin. Use Chrome DevTools (Inspect) to show live.

□ CSS Superpowers to Teach

- **Colors:** hex (#0ea5e9), rgb(), named
- **Flexbox:** display:flex; justify-content:center;
- **Grid:** display:grid; grid-template-columns: 1fr 1fr;
- **Typography:** font-size, font-weight, line-height
- **Spacing:** margin, padding (use rem/em)

□ Responsive Made Simple

Media Query Template:

```
@media (max-width: 768px) {
  .card { width: 100%; }
}
```

Teach mobile-first: design for phone, then desktop.

□ CLASS ACTIVITY (30 MIN)

Challenge: Style the about.html from last class. Add: blue header, centered card with white background, rounded profile image, styled button with hover effect. Use Flexbox to center.

Mini Competiton: Who makes the best looking page? Show 3 top designs on projector.

Module 3: JavaScript — The Brain

Duration: 2-3 Sessions | **Objective:** Make websites interactive and respond to users.

3.1 WHY JAVASCRIPT?

HTML = What you see, CSS = How it looks, **JavaScript = What it DOES**. Handles clicks, forms, animations, data, games, everything interactive.

3.2 WHERE JS RUNS

In the browser! No installation. Open any page → Right Click → Inspect → Console → you can run JS.

JAVASCRIPT - BASICS

```
// Variables
let name = "Amina";
const age = 16;
let isStudent = true;

// Functions
function greet(person) {
  return "Hello " + person + "!";
}

// Event - Button Click
document.getElementById("myButton").addEventListener("click", function() {
  alert(greet(name));
});
```

3.3 DOM MANIPULATION - THE MAGIC

DOM (Document Object Model) is how JS talks to HTML. You can change text, colors, hide/show, create elements.

JAVASCRIPT - LIVE EXAMPLE

```
<button id="btn">Change Color</button>
<h1 id="title">Skoolyn</h1>

<script>
  const btn = document.getElementById("btn");
  const title = document.getElementById("title");
  btn.addEventListener("click", () => {
    title.style.color = "#0ea5e9";
    title.textContent = "Smarter Learning!";
  });
</script>
```

Core Concepts to Cover

- Variables: let, const
- Data Types: String, Number, Boolean, Array, Object
- Conditions: if / else
- Loops: for, forEach
- Events: click, submit, input
- Functions: reusable code blocks

▢ Teacher Demo Ideas

- Counter App (+ / - buttons)
- Light/Dark Mode Toggle
- Simple Quiz: Check answer
- Form validation: Show error if empty

All demos <15 lines. Keep it simple, visual, instant feedback.

▢ CLASS ACTIVITY (35 MIN)

Build: Interactive Portfolio. Add a button "Contact Me" → opens alert with email. Add another button "Change Theme" → toggles background color using JS. Add live character counter for textarea.

Stretch: Play with localStorage to remember theme.

Module 4: Introduction to Backend

Duration: 2 Sessions | **Objective:** Understand what happens behind the scenes - where data is stored and how frontend talks to backend.

Big Picture: Until now, everything lives in the browser. Refresh and data is gone. Backend is the kitchen of a restaurant — customers (frontend) see the menu & tables, but cooking & storage happens in the kitchen (backend) unseen but essential.

4.1 FRONTEND VS BACKEND

FRONTEND (CLIENT)	BACKEND (SERVER)
What user sees: Browser	What user doesn't see: Server
HTML, CSS, JavaScript	Node.js, Python, PHP, Databases
Runs on user's phone/laptop	Runs on cloud server (e.g. AWS)
Portfolio website, Instagram UI	Saves likes, comments, login logic

4.2 HOW THE WEB WORKS (REQUEST → RESPONSE)

USER → Types skooyln.com → BROWSER → Sends REQUEST to SERVER → SERVER processes → Sends RESPONSE (HTML/CSS/JS) → BROWSER shows page

Teach terms: **Client** (browser making request), **Server** (computer always on), **HTTP** (language they speak), **Status Codes** (200 OK, 404 Not Found).

4.3 BACKEND LANGUAGES - SIMPLIFIED OVERVIEW

popular Choices

- **Node.js (JavaScript):** Same language as frontend. Great for beginners. We use this.
- **Python (Django/Flask):** Easy, readable, great for AI
- **PHP:** Powers WordPress, Facebook early

Databases (Where data lives)

- **SQL:** Tables like Excel (MySQL, PostgreSQL)
- **NoSQL:** Flexible docs (MongoDB, Firebase)

Example: All Instagram photos are in a database, not in HTML file.

4.4 YOUR FIRST BACKEND SERVER (NODE.JS + EXPRESS)

No need to master, just understand and run. Show magic in 10 lines.

BACKEND - SERVER.JS (NODE.JS + EXPRESS)

```
// Step 1: npm install express
const express = require("express");
const app = express();
```

```
// When someone visits /
app.get("/", (req, res) => {
  res.send("Hello from Skoolyn Backend! 🚀");
});

// API endpoint returning JSON
app.get("/api/students", (req, res) => {
  res.json([
    { name: "Amina", course: "Web Dev" },
    { name: "John", course: "Design" }
  ]);
});

app.listen(3000, () => console.log("Server running on http://localhost:3000"));
```

4.5 WHAT IS AN API?

API = Application Programming Interface. A waiter between frontend and backend. Frontend says "Give me student list" → API goes to database → returns data (usually JSON).

FRONTEND TALKING TO BACKEND - FETCH()

```
// In your HTML/JS file
fetch("http://localhost:3000/api/students")
  .then(response => response.json())
  .then(data => {
    console.log(data); // Shows student list!
    document.getElementById("list").innerHTML =
      data.map(s => `<li>${s.name} - ${s.course}</li>`).join("");
  });
```

📌 **Real Life Example:** When you open Twitter, frontend calls API `/api/tweets` to get tweets from database. Scroll → calls again. Like button → sends POST request to backend to save like.

4.6 SECURITY BASICS (TEACH EARLY!)

- Never share passwords/API keys in frontend code
- Validate data on backend (frontend validation can be bypassed)
- Use HTTPS (encrypted)
- Hash passwords (don't store plain)

Capstone Project: Skoolyn Portfolio + Backend

Students build and present final project integrating all modules.

Phase 1 - Frontend (Week 1-3)

- 1-page portfolio: About, Skills, Projects, Contact
- Semantic HTML5
- Styled with CSS Flexbox/Grid + Responsive
- JS interactions: theme toggle, form validation, typing animation

Phase 2 - Backend Connection (Week 4)

- Simple Node.js server serves portfolio
- /api/contact endpoint saves messages (in-memory or JSON file)
- Contact form uses fetch() to post to backend
- Deploy frontend on Netlify, backend on Render (free)

DEPLOYMENT CHECKLIST

STEP	ACTION	FREE TOOL
1	Push code to GitHub	github.com
2	Deploy frontend (HTML/CSS/JS)	Netlify / Vercel
3	Deploy backend (Node)	Render / Railway
4	Share live link	QR code for class

Assessment & Rubrics

Use this to grade fairly and give constructive feedback.

CRITERIA	EXCELLENT (5)	GOOD (3-4)	NEEDS WORK (1-2)	WEIGHT
HTML Structure	Semantic, valid, clean	Works but uses many divs	Broken tags, invalid	25%
CSS Design	Responsive, modern, flex/grid	Styled but not responsive	No styling / inline only	25%
JavaScript	Interactive, DOM, no errors	One interaction works	No JS or broken	25%
Backend Understanding	Can explain request/response & fetch API works	Understands concept but API fails	Cannot explain	15%
Presentation	Clear demo + answers Q&A	Reads slides, demo works	No demo	10%

TEACHER TIPS FOR SUCCESS

☐ Do

- Live code — don't just show slides
- Use VS Code + Live Server extension
- Praise effort, debug together
- Girls & boys equally — call everyone
- Use Skoolyn colors (#0ea5e9) for inspiration

☐ Don't

- Don't copy-paste without explaining
- Don't jump to frameworks (React) too early
- Don't shame errors — errors are learning
- Don't make students memorize — make them understand

GLOSSARY FOR STUDENTS

TERM	SIMPLE DEFINITION
HTML	Structure of webpage
CSS	Design and layout
JavaScript	Makes page interactive
Backend	Server + Database that stores data
Frontend	What users see in browser
API	Messenger between frontend & backend
DOM	Browser's map of HTML that JS can change
JSON	Data format: {"name":"Amina"}
Responsive	Works on phone & laptop

Ready to teach the future?

This resource is crafted for Skoolyn educators. Share with students, remix, and build.

SKOOLYN
Smarter Learning