Unit 2 HTML

Content Area: **Computer Science**
Course(s):
Time Period: **Marking Period 1**
Length: **20 days**
Status: **Published**

Summary

Unit 2 is where students move from understanding the web to building for it. Having learned in Unit 1 how web pages travel to a browser and how browsers interpret code, students now write that code themselves, using HTML (HyperText Markup Language) to structure the content of a web page. They begin with the idea of tags and elements, assemble a properly structured HTML document, and then progressively add and organize content: formatted text, hyperlinks, images, lists, and tables. Along the way they learn to use others' images and media responsibly through a dedicated lesson on copyright and attribution. The unit closes by introducing basic styling and color, which sets up the transition to CSS in Unit 3, and a summative quiz.

Because every lesson is hands-on, students are continually creating, testing, and debugging their own web pages, developing habits of careful structure, documentation, and iteration that carry through the rest of the course.

Revision Date: July 2026

Essential Questions

- How does HTML tell a browser what to display, and how do tags and elements structure a page?
- How do we add and organize different kinds of content — text, links, images, lists, and tables — on a web page?
- How do we use images and content created by others legally and ethically?
- How does styling change the way a page looks, and how can something like color be represented in code?

Enduring Understandings

- A website's HTML gives meaning and structure to content, separate from how it looks. Markup isn't decoration; it's the semantic skeleton browsers, assistive tech, and search engines use to interpret a page.
- A small set of structural elements (headings, paragraphs, lists, links, images, tables, forms) can represent almost any content. Picking the right element is a design decision, not a style choice.

- HTML is nested and hierarchical, and that hierarchy shapes how content gets read and navigated by people and machines alike.
- Semantic HTML makes a site accessible to all users, including those on screen readers, tying directly to the course's digital citizenship thread.
- The structuring skills learned here carry over to any interactive or dynamic system, making HTML a transferable entry point into computational thinking.
- Building a site is iterative: plan content, structure it, then test and revise; the same process students repeat in Unit 3 when they restyle this site with CSS.

Students Will Know

- HTML uses tags and elements to structure content, and the browser renders that markup into a visible page.
- Web pages are built from nested elements, and correct structure and nesting determine how a page displays.
- Different elements serve different purposes — headings and paragraphs for text, anchors for links, images for media, and lists and tables for organizing content.
- Using media created by others carries legal and ethical responsibilities, including respecting copyright and giving proper attribution.
- Styling separates how a page looks from what it contains, and a single property such as color can be represented in more than one way (names, hex, or RGB).

Students Will Be Skilled At

- Explaining what HTML is and how a browser renders tags and elements into a web page.
- Building a properly structured HTML page using the standard document skeleton with correctly nested and documented elements.
- Formatting text using headings, paragraphs, line breaks, and emphasis (bold and italic) tags.
- Creating hyperlinks using absolute and relative paths, and embed images with appropriate source and alt-text attributes.
- Organizing content using ordered lists, unordered lists, and tables.
- Explaining copyright, fair use, and Creative Commons, and give proper attribution when using others'

work.

- Applying basic styling and color (named, hexadecimal, and RGB) to HTML elements, previewing the transition to CSS.

Learning Plan

Lesson Sequence

Lesson	Focus	What Students Do
2.1 Introduction to HTML	Introduces HTML as the language that structures web content, and the concept of tags and elements.	Watch the lesson video and quiz; write first HTML using opening and closing tags; observe how the browser renders markup.
2.2 Structure of an HTML Page	The standard document skeleton and how elements nest inside one another.	Build a complete page with the doctype, html, head, title, and body elements; practice correct nesting, indentation, and comments.
2.3 Formatting Text	Structuring and emphasizing text.	Add headings (h1–h6), paragraphs, line breaks, and bold and italic emphasis to organize written content.
2.4 Creating Links	Hyperlinks and the anchor element.	Create links with the anchor tag and href attribute; distinguish absolute from relative links and link between pages.
2.5 Incorporating Images	Embedding images and making them accessible.	Add images with the img tag using src and alt attributes; adjust size and source images appropriately.
2.6 Using Lists	Organizing content in lists.	Build ordered and unordered lists with list items; nest lists to show hierarchy.
2.7 HTML Tables	Presenting information in rows and columns.	Construct tables using table, row, header, and data cells to organize tabular content.
2.8 Copyright	Using others' work legally and ethically.	Learn copyright, fair use, public domain, and Creative Commons; practice finding usable media and writing proper attributions.
2.9 Applying Styling	Changing how a page looks; a bridge to CSS.	Apply basic styling to elements and refine the appearance of a page, previewing the separation of content and style in Unit 3.
2.10 HTML Colors	Representing and applying color.	Apply color using named colors, hexadecimal codes, and RGB values; compare how the same color can be written different ways.
2.11 Quiz: Exploring	Summative assessment of the unit.	Complete the end-of-unit quiz

Web Design	demonstrating mastery of HTML structure, content elements, and copyright.

Key Vocabulary

Term	Definition
HTML	HyperText Markup Language — the language used to structure the content of a web page.
Tag	The markup, written in angle brackets, that defines the start or end of an element (e.g., and).
Element	A complete unit of content made of an opening tag, content, and a closing tag.
Attribute	Extra information added inside an opening tag that modifies an element (e.g., src, href, alt).
Nesting	Placing elements inside other elements; correct nesting determines how the page is structured.
Doctype / head / body	The document skeleton: the doctype declares the page type, the head holds page information, and the body holds visible content.
Heading (h1–h6)	Tags that create titles and subtitles, from most (h1) to least (h6) prominent.
Anchor (a) / href	The anchor element creates a hyperlink; its href attribute holds the destination address.
Absolute vs. relative link	An absolute link gives a full web address; a relative link points to a file within the same site.
Image (img) / src / alt	The img element embeds an image; src gives its location and alt provides descriptive text for accessibility.
Ordered / unordered list	Numbered (ol) or bulleted (ul) lists made of list items (li).
Table / row / cell	A table organizes content into rows (tr) and header (th) or data (td) cells.
Copyright	The legal right of a creator to control how their original work is used.
Fair use / Creative Commons	Fair use permits limited use of protected work; Creative Commons licenses let creators grant reuse rights in advance.
Attribution	Crediting the creator when you use their work.
Hex code / RGB	Two ways of representing a color in code — a six-digit hexadecimal value or red, green, and blue amounts.

Assessment

Formative

- Lesson videos with corresponding quizzes for each concept.

- Hands-on coding exercises in which students build and extend web pages (text, links, images, lists, tables, styling, and color).
- A copyright activity in which students locate usable media and write correct attributions.
- Peer code review of student pages, using the Peer Code Review template.
- Beginning- and end-of-class discussion questions.

Summative

- 2.11 Quiz: Exploring Web Design — the end-of-unit assessment covering HTML structure, content elements, and copyright.
- Unit 2 project: a student-designed website that demonstrates every HTML skill from the unit — see the Unit 2 Project section below.

Unit 2 Project: Build-Your-Own Website

The unit ends with a build-your-own website project. Students choose their own topic — a hobby, a favorite team, a cause they care about, a fictional world, a small "business," or a personal profile — and build a site about it. The topic is open so students take ownership of their work, but every project must demonstrate the full set of HTML skills taught in the unit. The result is a single artifact that pulls the whole unit together and a natural bridge into styling with CSS in Unit 3.

Project Requirements

Whatever the topic, each project must include all of the following. Students can use this as a checklist as they build.

HTML Skill	What the Project Must Include	Lesson
Page structure	A complete, valid HTML document — doctype, html, head with a title, and body — with comments documenting the code.	2.1–2.2
Formatted text	A heading hierarchy (an h1 plus at least one subheading), paragraphs, and bold and/or italic emphasis.	2.3
Links	At least two links: one to an external site (absolute) and one connecting the student's own pages or sections (relative).	2.4
Images	At least two images, each with descriptive alt text.	2.5
Lists	At least one ordered or unordered list.	2.6
Table	At least one table that organizes related information into rows and columns.	2.7
Copyright	Proper attribution for every borrowed image or media item, using openly licensed or public-domain sources.	2.8
Styling	Basic styling applied to page elements.	2.9
Color	Color applied using named colors, hexadecimal codes, or RGB values.	2.10

Suggested Process

- Plan — Choose a topic and sketch a simple layout, noting where each required element will go.
- Build — Create the page structure first, then add content one element at a time.
- Source media responsibly — Find openly licensed or public-domain images and record their attributions as you go.
- Test & debug — Preview in the browser often; confirm every element renders and every link works.
- Peer review — Exchange sites using the Peer Code Review template, then act on the feedback.
- Revise & share — Polish the site and present it to the class.

Deliverables & Timing

- The finished website (HTML in the CodeHS editor).
- An attribution list crediting all borrowed images and media.
- A short reflection describing the site and one thing the student would improve with more time.

Plan on roughly 4–6 class periods: one to plan, two to three to build, one for peer review and revision, and one to share. The project synthesizes the unit's Algorithms & Programming standards — decomposing the page (8.1.8.AP.4), incorporating and attributing media (8.1.8.AP.7), documenting code (8.1.8.AP.9), incorporating peer feedback (8.1.8.AP.6), and testing and refining the result (8.1.8.AP.8) — together with accessibility (8.1.8.IC.2).

Assessment Criteria

Criteria	What to Look For
Required elements	All nine required HTML features are present and used correctly.
Structure & code quality	The document is properly nested and commented, and the site renders correctly in the browser.
Content & organization	The topic is clear, and the content is logically organized and readable.
Copyright & attribution	All borrowed media is openly licensed or public domain and is properly credited.
Styling & design	Styling and color are applied intentionally to make the page clear and appealing.

Benchmark

• Mid-unit structured HTML skills check: working independently, students build a short page containing a

properly nested document skeleton, headings, paragraphs, a list, and a hyperlink, scored against a shared rubric aligned to 8.1.8.AP.4 and 8.1.8.AP.9. Results are compared across sections to gauge progress toward the unit's structure-and-nesting standards ahead of the culminating project.

Alternative

• Attribution portfolio and reflection: alongside the Peer Code Review, students compile a short portfolio entry — the images and media they sourced for their project with correct attributions — plus a written reflection on one HTML skill they still find challenging. This performance-based, nontest format lets students demonstrate their understanding of copyright and content organization outside a timed quiz.

Materials

Core instructional materials:

- CodeHS Web Design course — Unit 2 lessons, videos, and slide decks.
- The CodeHS HTML editor / sandbox for writing and previewing code.
- Student devices with internet access and a web browser.
- Image sources for the images and copyright lessons (e.g., Creative Commons Search, public-domain and openly licensed image libraries).
- Peer Code Review template and Class Discussion Notes template.

Supplemental materials: Optional printed references — an HTML tag "cheat sheet" and printed slides for students who benefit from them.

Standards

Standards Alignment

This unit addresses the following performance expectations from the 2020 NJSLS – Computer Science & Design Thinking (by the end of Grade 8). Because New Jersey treats building web pages as creating computational artifacts, HTML work aligns primarily to the Algorithms & Programming (AP) sub-concept.

Code	Performance Expectation	Lesson(s)
8.1.8.AP.4	Decompose problems and sub-problems into parts to facilitate the design, implementation, and review of programs.	2.1, 2.2, 2.3, 2.6, 2.7

8.1.8.AP.6	Refine a solution that meets users' needs by incorporating feedback from team members and users.	2.9
8.1.8.AP.7	Design programs, incorporating existing code, media, and libraries, and give attribution.	2.5, 2.8
8.1.8.AP.8	Systematically test and refine programs using a range of test cases and users.	2.4, 2.9
8.1.8.AP.9	Document programs in order to make them easier to follow, test, and debug.	2.2
8.1.8.CS.4	Systematically apply troubleshooting strategies to identify and resolve hardware and software problems in computing systems.	Throughout (debugging)
8.1.8.DA.1	Organize and transform data collected using computational tools to make it usable for a specific purpose.	2.7
8.1.8.DA.2	Explain the difference between how the computer stores data as bits and how the data is displayed.	2.10
8.1.8.IC.2	Describe issues of bias and accessibility in the design of existing technologies.	2.3, 2.5, 2.10

Computer Science & Design Thinking Practices

The following practices are developed throughout the unit:

- Practice 4: Developing and using abstractions — treating HTML tags and elements as reusable building blocks.
- Practice 5: Creating computational artifacts — building and publishing original web pages.
- Practice 6: Testing and refining computational artifacts — previewing, debugging, and improving pages.
- Practice 7: Communicating about computing and design — documenting code and giving attribution for borrowed media.

Note on Copyright & Media Literacy (NJSLS 9.4)

The Copyright lesson (2.8) is anchored in CS & Design Thinking through 8.1.8.AP.7 (giving attribution), but the fuller treatment of copyright, fair use, and Creative Commons also aligns to NJSLS 9.4 Career Readiness, Life Literacies & Key Skills — specifically the Information & Media Literacy strand. If your district maps intellectual-property and media-literacy content to 9.4, those performance expectations can be added alongside the CS codes above.

DA	Data & Analysis
	People use digital devices and tools to automate the collection, use, and transformation of data. The manner in which data is collected and transformed is influenced by the type of digital device(s) available and the intended use of the data.
8.1.8.DA.1	Organize and transform data collected using computational tools to make it usable for a specific purpose.

	Data is represented in many formats. Software tools translate the low-level representation of bits into a form understandable by individuals. Data is organized and accessible based on the application used to store it.
8.1.8.DA.2	Explain the difference between how the computer stores data as bits and how the data is displayed.
8.1.8.DA.3	Identify the appropriate tool to access data based on its file format.
	Computer models can be used to simulate events, examine theories and inferences, or make predictions.
8.1.8.DA.5	Test, analyze, and refine computational models.
8.1.8.DA.6	Analyze climate change computational models and propose refinements.
AP	Algorithms & Programming
	Individuals design algorithms that are reusable in many situations. Algorithms that are readable are easier to follow, test, and debug.
8.1.8.AP.1	Design and illustrate algorithms that solve complex problems using flowcharts and/or pseudocode.
	Programmers create variables to store data values of different types and perform appropriate operations on their values.
8.1.8.AP.2	Create clearly named variables that represent different data types and perform operations on their values.
	Control structures are selected and combined in programs to solve more complex problems.
8.1.8.AP.3	Design and iteratively develop programs that combine control structures, including nested loops and compound conditionals.
	Programs use procedures to organize code and hide implementation details. Procedures can be repurposed in new programs. Defining parameters for procedures can generalize behavior and increase reusability.
8.1.8.AP.4	Decompose problems and sub-problems into parts to facilitate the design, implementation, and review of programs.
8.1.8.AP.5	Create procedures with parameters to organize code and make it easier to reuse.
	Individuals design and test solutions to identify problems taking into consideration the diverse needs of the users and the community.
8.1.8.AP.6	Refine a solution that meets users' needs by incorporating feedback from team members and users.
8.1.8.AP.7	Design programs, incorporating existing code, media, and libraries, and give attribution.
8.1.8.AP.8	Systematically test and refine programs using a range of test cases and users.
8.1.8.AP.9	Document programs in order to make them easier to follow, test, and debug.
	Engineering design is a systematic, creative, and iterative process used to address local and global problems. The process includes generating ideas, choosing the best solution, and making, testing, and redesigning models or prototypes.
8.2.8.ED.1	Evaluate the function, value, and aesthetics of a technological product or system, from the perspective of the user and the producer.
8.2.8.ED.2	Identify the steps in the design process that could be used to solve a problem.
8.2.8.ED.3	Develop a proposal for a solution to a real-world problem that includes a model (e.g., physical prototype, graphical/technical sketch).
8.2.8.ED.4	Investigate a malfunctioning system, identify its impact, and explain the step-by-step process used to troubleshoot, evaluate, and test options to repair the product in a

	collaborative team.
	Engineering design requirements and specifications involve making trade-offs between competing requirements and desired design features.
8.2.8.ED.5	Explain the need for optimization in a design process.
8.2.8.ED.6	Analyze how trade-offs can impact the design of a product.
8.2.8.ED.7	Design a product to address a real-world problem and document the iterative design process, including decisions made as a result of specific constraints and trade-offs (e.g., annotated sketches).

Suggested Strategies for Modification

Suggested Strategies for Modification for Computer Science