Unit 3 CSS

Content Area: **Computer Science**
Course(s):
Time Period: **Marking Period 1**
Length: **14**
Status: **Published**

Brief Summary of Unit

Unit 3 completes the arc of the course: after learning how the web works (Unit 1) and how to build a page's content with HTML (Unit 2), students now learn to control how that content looks using CSS (Cascading Style Sheets). The unit's central idea is the separation of concerns — HTML defines content and structure, while CSS defines presentation. Students learn to write CSS rules and to target exactly the elements they want to style using three kinds of selectors: by tag, by class, and by ID. They then learn how the cascade resolves conflicts when rules compete. A scaffolded “Global Trip” project runs through the first four lessons, and the unit ends with students styling their own pages and a summative quiz.

The unit is highly hands-on: nearly every lesson is built around short styling exercises, so students are constantly writing, previewing, and refining CSS. It culminates in a project in which students return to the website they built in Unit 2 and give it a full CSS makeover.

Revision Date: July 2026

Essential Questions

- Why do we separate a page's content (HTML) from its presentation (CSS)?
- How do CSS selectors let us target exactly the elements we want to style?
- When two style rules conflict, how does the browser decide which one wins?
- How does styling make a page not just look better, but also clearer and more readable?

Enduring Understandings

- CSS controls how a page looks, keeping presentation separate from the content and structure defined by HTML.

- A CSS rule pairs a selector (what to style) with declarations (how to style it).
- Selectors range from broad (by tag) to reusable (by class) to specific (by ID), and the right choice depends on the goal.
- When rules conflict, the cascade — based on specificity, order, and importance — determines the outcome.
- Thoughtful styling improves not just appearance but readability and accessibility.

Students Will Know

- What CSS is and why separating presentation from content is valuable.
- How to write CSS rules using correct selector-and-declaration syntax.
- How to select and style elements by tag, by class, and by ID, choosing the appropriate selector for the goal.
- How to predict and control which rule applies when styles conflict, using the cascade, specificity, and importance.
- How to apply CSS to their own web pages to create a clear, readable, and visually appealing design.

Learning Plan

Lesson Sequence

A scaffolded “Global Trip” project runs through lessons 4.1–4.4 (skeleton → style headers → emphasize with styles → add photos), giving students a guided site to build up before they style their own.

Lesson	Focus	What Students Do
4.1 Introduction to CSS Styling	What CSS is, why we separate style from content, and basic rule syntax.	Watch the intro video and quiz; complete Styling with CSS, Styling a List, and Adding CSS Styling; set up the Global Trip Skeleton.
4.2 CSS Select by Tag	Styling every element of a given type.	Learn tag (type) selectors; practice with Rainbow, Puppy Styling, Today's Top Videos, and World Heritage Sites; style headers in Global Trip.
4.3 CSS Select by Class	Reusable styles applied to groups of elements.	Learn class selectors; practice with Simple Checkerboard, Football

		Divisions, and Favorite Navigation; emphasize elements with classes in Global Trip.
4.4 CSS Select by ID	Unique styling for a single element.	Learn ID selectors; practice with Logo, Water Cycle, and Aquarium Features; add photos to Global Trip; earn the Selector Badge.
4.5 The Cascade	How the browser resolves conflicting rules.	Learn specificity, order of precedence, and importance; practice with Using Importance, Order of Precedence, Planning for the Weekend, Highlight to Remember, and Top Theme Parks.
4.6 Add CSS Styling to your Homepage	Applying CSS to their own page.	Style their own homepage, pulling together tag, class, and ID selectors and the cascade — the launch point for the unit project.
4.7 CSS – Styling Websites Quiz	Summative assessment of the unit.	Complete the end-of-unit quiz covering CSS syntax, the three selector types, and the cascade.

Assessment

Assessment

Formative

- Lesson quizzes for 4.1 through 4.5.
- The many short styling exercises in each lesson (e.g., Rainbow, Puppy Styling, Simple Checkerboard, Water Cycle, Top Theme Parks).
- The scaffolded Global Trip project built across lessons 4.1–4.4, and the Selector Badge (4.4.7).
- Peer code review of student-styled pages.
- Beginning- and end-of-class discussion questions.

Summative

- 4.7 CSS – Styling Websites Quiz — the end-of-unit assessment covering syntax, selectors, and the cascade.
- Unit 3 project: a full CSS restyle of the student's Unit 2 website — see the Unit 3 Project section below.

Unit 3 Project: Restyle Your Website with CSS

For the final project, students return to the website they built in the Unit 2 HTML project and give it a complete visual makeover using CSS. The requirement that makes this a true test of the unit: students must first remove all of the old inline styling from their Unit 2 site — the style attributes and inline colors added in the HTML unit — and then rebuild every bit of that styling as proper CSS. Doing so demonstrates the big idea of the unit, separating content (HTML) from presentation (CSS), and gives each student a clear before-and-after of the same site.

Project Requirements

Working from their own Unit 2 site, each project must include all of the following:

CSS Skill	What the Project Must Include	Lesson
Separation of concerns	Remove every inline style attribute and inline color from the Unit 2 site so the HTML holds only content and structure.	4.1
Where CSS lives	Put all styling in one organized place — an external stylesheet or a single style block — rather than scattered inline.	4.1
Select by tag	Use at least two type (tag) selectors to style all elements of a kind, such as every paragraph or every heading.	4.2
Select by class	Create and apply at least one class to style a group of elements consistently.	4.3
Select by ID	Use at least one ID selector to uniquely style a single element.	4.4
The cascade	Include at least one place where rules overlap, and note which rule wins and why.	4.5
Core properties	Apply color, background, text/font styling, and spacing so the whole page looks intentionally designed.	4.1–4.6
Readability	Keep text legible with good color contrast.	Accessibility

Suggested Process

- Start from your Unit 2 site — Open the website you built for the HTML project.
- Strip the old styling — Remove the inline style attributes and inline color; confirm the page still shows all its content, now unstyled.
- Plan your selectors — Decide what to style by tag, by class, and by ID.
- Write your CSS — Build the stylesheet, previewing in the browser as you go.
- Use the cascade on purpose — Where rules compete, let specificity and order settle it intentionally.
- Peer review & polish — Swap sites, gather feedback, and refine.

Deliverables & Timing

- The restyled website: HTML containing only content and structure, plus a CSS stylesheet.
- A short before-and-after reflection describing what changed and giving one example of the cascade in action (which rule won and why).

Plan on roughly 4–6 class periods: one to remove the old styling and plan, two to three to write the CSS, one for peer review and revision, and one to share. The project synthesizes the unit's standards — organizing and decomposing the styles (8.1.8.AP.4), incorporating feedback (8.1.8.AP.6), documenting the stylesheet (8.1.8.AP.9), troubleshooting conflicting rules (8.1.8.CS.4), and testing and refining the result (8.1.8.AP.8) — along with readability and accessibility (8.1.8.IC.2).

Assessment Criteria

Criteria	What to Look For
Separation of concerns	The HTML holds only content and structure; all styling lives in CSS.
Selector variety	Tag, class, and ID selectors are each used appropriately.
The cascade	At least one rule conflict is handled intentionally, and the student can explain which rule wins.
Core styling	Color, text, and spacing are applied thoughtfully across the whole page.
Readability & polish	Text is legible with good contrast, and the design looks clean and intentional.

Benchmark

• Mid-unit selector-and-cascade skills check: working independently, students are given a short unstyled page and a set of competing CSS rules, then must correctly target elements by tag, class, and ID and predict which rule the cascade will apply, scored against a shared rubric aligned to 8.1.8.AP.4 and 8.1.8.CS.4. Results are compared across sections to gauge progress toward the unit’s selector-and-cascade standards ahead of the culminating restyle project.

Alternative

• Before-and-after portfolio reflection: alongside the Unit 3 project’s required reflection, students present a short portfolio pairing a screenshot of their old inline-styled Unit 2 page with their new CSS-styled page and explain, in their own words, one example of the cascade in action. This performance-based format lets students demonstrate mastery of separating content from presentation without a timed written test.

Materials

Core instructional materials:

- CodeHS Web Design course — the CSS lessons (Module 4), videos, and slide decks.
- The CodeHS editor for writing and previewing CSS.
- Each student's Unit 2 project files — needed as the starting point for the restyle project.
- Student devices with internet access and a web browser.
- A color-contrast checker (optional) to support readable, accessible color choices.
- Peer Code Review template and Class Discussion Notes template.

Supplemental materials: Optional printed references — a CSS property / selector "cheat sheet" and printed slides.

Standards

Standards Alignment

This unit addresses the following performance expectations from the 2020 NJSLS – Computer Science & Design Thinking (by the end of Grade 8). As with the HTML unit, styling web pages is treated as creating and refining computational artifacts, so the 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.	4.1, 4.6, Project
8.1.8.AP.6	Refine a solution that meets users' needs by incorporating feedback from team members and users.	4.6, Project
8.1.8.AP.7	Design programs, incorporating existing code, media, and libraries, and give attribution.	4.4
8.1.8.AP.8	Systematically test and refine programs using a range of test cases and users.	4.5, 4.6, Project
8.1.8.AP.9	Document programs in order to make them easier to follow, test, and debug.	Throughout (CSS organization)
8.1.8.CS.4	Systematically apply troubleshooting strategies to identify and resolve hardware and software problems in computing systems.	4.5, Throughout
8.1.8.IC.2	Describe issues of bias and accessibility in the design of existing	4.1, 4.6, Project

technologies.

Computer Science & Design Thinking Practices

The following practices are developed throughout the unit:

- Practice 4: Developing and using abstractions — classes and selectors as reusable style abstractions applied across many elements.
- Practice 5: Creating computational artifacts — styling and refining web pages.
- Practice 6: Testing and refining computational artifacts — previewing styles and resolving conflicts through the cascade.
- Practice 7: Communicating about computing and design — writing organized, documented stylesheets.

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.

Suggested Strategies for Modification

Suggested Strategies for Modification for Computer Science