

Lesson Details Guidance

- **Unit:** Indicate the unit to which the lesson belongs to
-

- **Lesson Number:** Assign a sequential number to the lesson within the unit for easy navigation and reference.
-

- **Teaching Methodologies:** Identify the instructional strategies employed in the lesson to enhance learning effectiveness
 - **Select Applicable Methods:** Choose from methodologies such as UbD (Understanding by Design), Gamification, Project-Based Learning, Flipped Classroom.
 - **Multiple Methods:** You can select more than one methodology if applicable.
 - **Align with Objectives:** Ensure the methodologies support the lesson's learning goals.

Methodology Descriptions:

- **Understanding by Design (UbD):** A framework for designing curriculum units, performance assessments, and instruction that lead students to deep understanding of the content.
 - **Gamification:** Incorporating game elements (points, badges, leaderboards) to motivate and engage learners.
 - **Project-Based Learning:** Students acquire deeper knowledge through active exploration of real-world challenges and problems.
 - **Flipped Classroom:** Students review content at home (videos, readings) and engage in interactive activities in class.
-

- **Lesson Duration (Minutes):** Specify the estimated time required to complete the lesson, aiding in planning and pacing.
 - **Default Duration:** Use 60 minutes as the standard unless adjustments are necessary.
-

- **Lesson Vocabulary Terms:** List key terms and concepts introduced or emphasized in the lesson to support vocabulary development.
 - **Select Relevant Terms:** Include important words that are crucial for understanding the lesson content.

***Example: *"*Lesson Vocabulary Terms:*

HTML (HyperText Markup Language), Element, Tag, Attribute, Syntax"

-
- **Lesson Resources / Materials:** Identify all materials and resources needed to effectively deliver the lesson.
 - **List All Necessary Materials:** Include slides, videos, readings, software, tools, and any handouts.
 - **Provide Access Information:** Include links, filenames, or instructions on how to access each resource.
 - **Ensure Relevance:** Resources should directly support the lesson objectives.

Example: "Lesson Resources / Materials:

- Slide Deck: "Introduction to HTML Structure"
- Video Tutorial: "Building Your First Webpage" (Link)
- Interactive Online Tool: CodePen for live coding practice
- Handout: "Common HTML Tags Reference Sheet" (PDF)
- Quiz: "HTML Basics Self-Assessment" (Accessible on learning platform)"

Stage 1: Desired Results

- **Learning Goals:**
 - Match verbs to the complexity of your objectives, starting with simpler ones like "Identify" and advancing to "Analyze" or "Create."
 - Not all levels are needed; focus on those relevant to your goals, such as understanding, applying, or creating.
 - it's enough to use: 1–2 levels for kids (6–9 years old) / 2–3 levels for ages (10–14) / 2–4 levels for ages (+16)

That said, it also depends on the complexity of the lesson. For +16 age groups, you might focus on a single Bloom's level if the concept is particularly complicated or important. This ensures depth in understanding critical concepts while keeping it manageable.

Category	Guidance	Verbs to Use (More Verbs)	Examples
Remembering (Knowledge Acquisition)	Identify goals focusing on recall of facts and basic concepts.	Define, Identify, List, Recall, Recognize	- Define what an HTML tag is.- Identify the main function.- List the basic HTML tags.- Recall the steps to run a script.- Recognize a for loop in the given code.
Understanding (Comprehension)	Set goals that require explaining ideas or concepts.	Explain, Describe, Summarize, Distinguish, Predict	- Explain how a CSS selector applies style to an element.- Describe what this function returns.- Summarize the purpose of the given algorithm.- Distinguish between a list and a dictionary.- Predict what the code will print when executed.
Applying (Skill Application)	Encourage applying knowledge in new situations.	Apply, Demonstrate, Modify, Solve, Use, Write	- Apply a loop to automate this task.- Demonstrate how to connect to a database.- Modify the function so it returns a sorted list.- Solve this bug in the code.- Use the built-in method to parse the data.- Write a script that reads a file and outputs its contents.
Analyzing (Critical Thinking)	Guide students to analyze and break down information.	Analyze, Compare, Contrast, Examine	- Analyze the algorithm's time complexity.- Compare the efficiency of the two sorting algorithms.- Contrast using recursion vs. iteration for this problem.- Examine the code to find why it's throwing an error.
Evaluating (Critical Thinking)	Develop goals that involve evaluating information and making judgments.	Evaluate, Judge, Justify, Critique, Assess, Test	- Evaluate the performance of the new feature.- Judge whether this approach is optimal.- Justify the choice of data structure.- Critique the code's readability and maintainability.- Assess whether this code meets the project requirements.- Test the function with various inputs.
Creating (Innovation)	Inspire students to create new or original work.	Create, Design, Develop, Construct, Generate	- Create a dashboard to display user metrics.- Design a class structure for the application.- Develop a user authentication module.- Construct a data


Category	Guidance	Verbs to Use (More Verbs)	Examples
			pipeline.- Generate a report from this dataset.

Understandings and Essential Questions:

Category	Enduring Understandings	Essential Question
Guidance	Enduring Understandings represent the core concepts, principles, or insights that students should retain long after the lesson ends. They address the “big ideas” of the discipline that are transferable and valuable in other contexts.	Essential Questions are open-ended, thought-provoking prompts that spark inquiry, encourage exploration, and guide students toward deeper understanding. They don’t have one “right” answer but invite multiple perspectives and ongoing investigation. Tips for Crafting Essential Questions:- Open-Ended Inquiry: Avoid yes/no questions; instead, prompt reflection and discussion.- Depth and Complexity: Encourage learners to go beyond surface-level understanding and grapple with underlying principles or complexities.- Real-World Relevance: Connect questions to real-life coding scenarios, broader societal issues, or students’ personal experiences.- Alignment with Objectives: Ensure that each question supports the learning goals and aligns with the Enduring Understandings.- Inquiry Encouragement: Pose questions that spark curiosity and the desire to learn more.- Stimulate Further Exploration: The best questions leave students wanting to investigate, test ideas, and discover more on their own.
Examples	- Understanding HTML is essential for web development and structuring content online.- The use of semantic HTML enhances the	- How does the structure of a webpage’s code affect the user experience?- Why is clarity and organization in code important when working on large projects?- How do the choices we make in code (e.g., naming variables, using comments,

Category	Enduring Understandings	Essential Question
	accessibility and search-ability of websites.	structuring functions) impact others who will read or maintain it?

Stage 2: Evidence

 Stage 2 focuses on the assessments and evidence teachers will collect to ensure that learners have met the learning goals, gained the intended understandings, and can answer the essential questions established in Stage 1. The evidence should be directly aligned with the desired results, allowing teachers to confidently measure student mastery and progress.

- **Performance Tasks: (Required at the end of the lesson)**

Category	Performance Task	Criteria/Rubrics
Guidance	- Design authentic tasks that challenge students to apply their learning in new and meaningful contexts.- Tasks should reflect real-world scenarios relevant to the subject matter.- Align tasks directly with the learning goals from Stage 1 and encourage higher-order thinking (e.g., applying, analyzing, evaluating, creating).	- Develop clear criteria or rubrics to evaluate student performance.- Criteria should be transparent, measurable, and directly aligned with the learning objectives.- Use rubrics that consider multiple dimensions of performance, such as accuracy, creativity, efficiency, clarity, or code quality.
Examples	Students create a responsive webpage that includes semantic HTML and organized CSS.	A rubric that evaluates code structure, use of semantic tags, effective styling, and responsiveness on different devices.

Instructional Strategies for Performance Tasks:

- **Differentiation:**
 - Adjust tasks for diverse learning styles (Visual, Auditory, Reading/Writing, Kinesthetic) and proficiency levels.
 - Offer multiple means of representation (e.g., code examples, visual diagrams, step-by-step tutorials).
 - Example: Use color-coded comments in code for visual learners, record a voice-over explanation for auditory learners, or provide a step-by-step worksheet for reading/writing learners.
- **Support:**
 - Provide scaffolding for students who need more guidance, such as hint cards, targeted tutoring, or simplified starter code.
 - Example: Offer a “starter template” with basic HTML structure for students who need extra help.
- **On-Level Instruction:**
 - Ensure the main activity is accessible at grade-level standards or skill expectations, offering a clear baseline of achievement.
- **Advanced Options:**
 - Present extension opportunities for advanced learners who have mastered the essential skills, such as adding interactivity with JavaScript or implementing a more complex layout.
 - Example: Challenge students to incorporate a responsive navigation menu or CSS animations.
- **Other Evidence**



These can include traditional assessments and informal checks that support and corroborate the evidence gathered from performance tasks. The goal is to provide a well-rounded picture of student understanding throughout the learning process.

Category	Quizzes/Tests/Peer Assessment (Optional)	Observations (Required)	Class Discussions (Required)
Guidance	- Traditional quizzes or short coding challenges to quickly gauge factual recall or basic skill application.- Peer reviews where students comment on each other's code can also provide insight into their understanding.	- Note how students approach problem-solving during hands-on activities.- Monitor engagement, collaboration, and the ability to debug or explain their code.	- Facilitate discussions where students explain their coding decisions, reflect on their design choices, or troubleshoot issues together.- Student contributions in discussions can reveal depth of understanding, communication skills, and critical thinking.
Example	A quick multiple-choice quiz on HTML tags and their purposes.	The instructor notes which students ask relevant debugging questions or can explain their reasoning process.	Students discuss how different coding approaches can solve the same problem and debate which solution is more efficient or readable.

Stage 3: Plan Learning Experiences and Instruction:

Pacing	Teacher / Student Actions	Materials
Engage (up to 5 min)		

- **Engage (up to 5min):**
 1. **Create Curiosity:**
 - Begin with a surprising fact, striking visual, or intriguing question that directly relates to the session's topic. This unexpected element should immediately spark learners' curiosity and make them want to learn more.
 2. **Grab Attention:**

- Follow up with a brief real-world example, relatable story, or a compelling scenario. This step cements the initial curiosity by demonstrating why the topic matters and how it connects to learners' lives or interests.

3. Frame the Lesson's Purpose

- Clearly state the main challenge or goal of the lesson. Explain what the learners will explore, solve, or achieve by the end. This ensures they understand both the relevance and the scope of the session, setting a clear path for further exploration.

Points Incentive: (+2 points for attendance, Additional +2 points for on-time attendance)

- **Explore (up to 5min):**

In this phase, learners actively engage with the topic through hands-on activities, investigations, or self-guided inquiries, allowing them to interact with the content, make connections, and foster curiosity and discovery.

- **Explain(up to 10-15min):**

Here, learners synthesize their findings and the instructor provides clear explanations or key concepts to enhance understanding.

- **Elaborate / Extend (up to 10min):**

In this phase, learners apply their knowledge to new situations, deepening their understanding and building connections.

- **Evaluate / Performance Task:**

This phase involves assessing learners' understanding and skills, often through activities that encourage reflection and feedback.

***Points Incentive:** Assign +10 points ***-*distributed based on a rubric- for successfully completing the performance task.*

- **Wrap up:**

- **Homework (When Needed):** Assign purposeful homework when necessary, accompanied by clear rubrics to guide expectations and ensure transparency in assessment.

Points Incentive: Award +10 points *-distributed based on a rubric- for completing homework with an additional +5 bonus points for on-time submissions.*