Lesson 2: Intro to HTML

45 minutes

Overview

This lesson introduces many new concepts and tools to students: they are introduced to HTML, the Web Lab tool, and how to navigate lesson resources on Code.org in general. In this lesson, students are introduced to HTML as a solution to the problem of how to communicate both the content and structure of a website to a computer. The lesson begins with a brief unplugged activity demonstrating the challenges of effectively communicating the structure of a web page. Students then look at an exemplar HTML page in Web Lab and discuss with their classmates how HTML tags help solve this problem. Students then write their first HTML. A wrap-up discussion helps to solidify the understanding of content vs. structure that was developed throughout the lesson.

Question of the Day: How can we tell the computer both *what* to put on the web page, and *how* to organize it?

Standards

Full Course Alignment

CSTA K-12 Computer Science Standards (2017)

▶ **AP** - Algorithms & Programming

Agenda

Warm Up (5 minutes)
- **The Need for HTML**

Activity (35 minutes)
- **Pair Programming**
- **Exploring HTML**

Wrap Up (5 minutes)
- **Reflection**

Objectives

Students will be able to:

- Explain that HTML allows a programmer to communicate the way content should be structured on a web page
- Understand how to use lesson resources provided in Web Lab
- Write a simple HTML document that uses opening and closing tags to structure content

Preparation

- Review the Code Studio levels
- Check the **"Teacher's Lounge"** forum for verified teachers to find additional strategies or resources shared by fellow teachers
- If you are teaching virtually, consider checking our **Virtual Lesson Modifications**

Links

Heads Up! Please make a copy of any documents you plan to share with students.

For the teachers

- **HTML Tags** - Resource

For the students

- **Intro to Web Lab Part 2** - Video (**Download**)
- **Pair Programming** - Video

Vocabulary

- **HTML** - Hypertext Markup Language, a language used to create web pages
- **HTML Element** - A piece of a website, marked by a start tag and often closed with an end tag
- **HTML Tag** - The special set of characters that indicates the start and end of an HTML element and that element's type
- **Website Content** - the text and images on a website
- **Website Structure** - how the content of a website is organized

Introduced Code

- `<!DOCTYPE>`
- `<body></body>`
- `<head></head>`
- `<html></html>`
- `<p></p>`

Teaching Guide

Warm Up (5 minutes)

The Need for HTML

Display: Show the image inside the **Exemplar Text Website**

Teaching Tip

If this site is blocked for students, your IT department may need to whitelist codeprojects.org. This is the same site that students will use to publish their own web pages, so it's important that they have access.

Prompt: *How could you explain to someone over the phone how to draw the following web page?*

Discuss: Once students have written their instructions, have them briefly share their instructions with a neighbor.

Discussion Goal: Activities like this one are often used in CS courses to help highlight just how much precision is needed to communicate instructions to a computer. In this instance the goal is similar. You want to highlight the challenge of differentiating the actual content on the page and instructions indicating how it should be structured. This demonstration helps justify the creation of HTML in order to tag pieces of content to help the computer understand what they are and hence how they should look.

Demo: Run a quick demo using the instructions below.

- Pick one student to verbally share one of their instructions with you.
- The teacher should act as the person on the phone trying to draw the web page
- Publicly “draw” the website exactly as the student's instructions say. For example, if told to "Write bigger", write the word "bigger" on the page. If they don't indicate where text goes then place text in random locations.
- As the student gives you instructions have them tell you if you have drawn it correctly. If you have not drawn it correctly, have them make their directions more specific until you can draw it correctly.
- Change students after a couple of instructions to get more students involved.
- Keep track of the instructions students give and the improvements they make to the instructions somewhere visible as well.
- Repeat this process until you have recreated most of the web page.

Discuss: Once you have finished drawing the site, quickly create a list of all the different kinds of information they needed to account for in their instructions. For example, location, size, font, etc.

Remarks

> There's a lot of information that we need to communicate if we want to create web pages. It's not enough to just know what content you want to put on your page, like the actual words or images. You need to know where things should be and how they should look. Today we're going to start learning the languages used on the web to represent this additional information.

Question of the Day: How can we tell the computer both *what* to put on the web page, and *how* to organize it?

Key Vocabulary:

- **website content** - the text and images on a website
- **website structure** - how the content of a website is organized

Activity (35 minutes)

Pair Programming

Group: Put students into pairs.

Remarks

> Today we're going to start using a new tool. We're going to explore this tool using something called **pair programming**. Pair programming helps people make better programs by working together, but there are some rules we have to follow to make sure it goes well.

Video: Show students the **Pair Programming** video in the slides.

Questions to consider with the video:

- Why do you think professional programmers use pair programming?
- How do you think pair programming will help you to program better?

Discussion Goal: The goals of this discussion center less around particular answers to this question and more around promoting positive attitudes toward pair programming. As students discuss its potential benefits, make sure they understand that this is an industry-standard practice, not just something fun to do.

Teaching Tip

Videos are used throughout the curriculum to spark discussions, supplement key concepts with additional explanations and examples, and expose students to the various roles and backgrounds of individuals in computer science.

While interacting with the video, turn on closed captioning so students can also read along as they watch.

To encourage active engagement and reflection, use one or more of the strategies discussed in the **Guide to Curriculum Videos**.

Review: Ensure that students understand the rules for pair programming:

- There is only one computer.
- The driver is the only one to touch the keyboard/mouse.
- The navigator should look for problems in the code and keep track of the high-level plan.
- Both driver and navigator should be communicating constantly.
- Driver and navigator must switch when the teacher indicates, typically every few minutes.

Teaching Tip

Practicing Communication: Some classes may need more support in communicating and collaborating effectively. If appropriate, consider having your students brainstorm a list of "sentence stems" that they can use for respectful and effective communication before they break into pairs ("Have you considered..." "What about..." "I think the problem might be..."). As students move through the lesson, be attentive to how students are working together - understanding the norms of pair programming is just as important as learning the new HTML code in this lesson!

To read more about pair programming, see the **Guide to Teaching and Learning Strategies**.

Exploring HTML

Transition: Have pairs go to Code Studio and both log in using the "Pair Programming" feature.

Display: Use the slide to guide your students on how to connect Pair Programming in Code Studio.

- Choose one partner to log into Code Studio.
- Click on your name, and choose "Pair Programming from the menu.
- Choose your partner's name from the list.
- Start coding as a team!

Do This: Remind students to switch driver and navigator every three minutes. You may want to project a digital timer at the front of the room.

1 **Exploration**

Teaching Tip

Facilitating Exploration Levels: Exploration levels are a great opportunity for students to think critically about code and engage in class discussions. Consider having students discuss with a partner before and after typing in code. Once they have had a chance to explore the code and discuss with a partner, bring the class together for a full-group discussion to discuss what they did in the Exploration level, what they **noticed**, and what they still **wonder**. Use this as an opportunity to address any misconceptions that students may have had about the code initially.

Digging Deeper: For more tips about programming levels, see the **CSD Guide to Programming Levels**. This document includes strategies and best-practices for facilitating programming levels with students.

Circulate: Give students time to explore the tool and complete the task on the level. Students may also discover different tabs and buttons on the page.

Prompt: *What did you notice about the workspace and the preview? What other features did you discover in this tool?*

Discussion Goal: Students should notice that even if they type sentences on multiple lines, the preview will show all of the sentences on a single line. Also highlight any additional features students discover in this tool and make sure all students are aware of it

Teaching Tip

Using Resources: Below you can find recommendations for using the many resources students are introduced to in the lesson. You could consider creating a "Resource Chart" to keep track of these options and support students to be self-sufficient as they progress through levels.

- **Level Instructions:** Instructions may introduce small pieces of new content. Each level features a "Do This" section explaining what students are supposed to do in that level. Set the expectation early that reading these instructions, not just the "Do This" section, is important.
- **Help and Tips Section** This tab contains all of the relevant videos and reference guides for a particular level.

Code Studio: Have students continue to the next level in Code Studio

2 Exploration

Teaching Tip

Using the Tool As students explore the site, make sure that they understand how to use the AI Chat tool and the "Help and Tips" section, which will give them access to the previous video.

Text-to-Speech Options: The instructions panel includes two options that can support comprehension for students.

- **Text to Speech** which reads aloud the instructions for students
- **Microsoft Immersive Reader** which opens a new panel for the instructions and gives controls to change the text size, contrast, or translate to another language.

Click here to learn more about these options

Regroup: Bring students back together once they've spent a couple of minutes looking through this level. The discussion prompts listed in the level should be used in a standard Think-Pair-Share structure.

- What code makes the text bigger and bolder?
- What code makes text appear as a list?
- What code makes the text appear on separate lines?
- What's a piece of code that doesn't appear to do anything on the screen?

It's okay to not address these questions as a full group or answer them completely - they will be introduced in the following video and covered in-depth throughout the unit. Instead, the goal of the discussion is to call out the features of HTML that students are noticing. The two primary takeaways (reinforced in the subsequent video as well) are that HTML uses a system of tags to surround content and indicate what it is and how it should be displayed.

Video: Show students the **Intro to Web Lab Part 2** video in the slides

Questions to consider with the video:

- Why are HTML tags useful?
- What does the paragraph tag do?

Discussion Goals: As students discuss HTML tags, make sure they understand that HTML tags are used to structure, or organize, content on the screen. Talking about the organization, structure, or role of the content in the page (heading, paragraph, list, etc.) is more accurate than talking about specific aspects of its appearance (such as size or spacing).

For the second prompt, there's a direct answer: the paragraph tag separates text into paragraphs. You may want to follow up this question by asking students how they think a web browser for the blind might deal with paragraphs. For example, while sighted people may use spacing and new lines to separate out paragraphs, what should a screen reader do?

Key Vocabulary:

- HTML - Hypertext Markup Language, a language used to create web pages
- HTML Element - A piece of a website, marked by a start tag and often closed with an end tag
- HTML Tag - The special set of characters that indicates the start and end of an HTML element and that element's type

Code Studio: Have students continue to the next level in Code Studio

3-5 **Skill Building**

3 4 5

Teaching Tip

Facilitating Skill Building Levels: Skill Building levels are designed to continue teaching new skills and blocks through exploration, trial-and-error, and using worked examples from pre-supplied code. Students are still getting familiar with the concepts in the lesson and will need strong support throughout these levels to build confidence, debug their code, and cement their understanding.

Consider having students complete Skill Building levels in pairs using **Pair Programming**, which has students use one computer and trade between being a Driver or a Navigator. This process is highlighted in **this video**, which you can show to the class. You can have students switch roles based on a timer, or switch every time they complete a level.

Digging Deeper: For more tips about programming levels, see the **CSD Guide to Programming Levels**. This document includes strategies and best-practices for facilitating programming levels with students.

6 **Practice**

Teaching Tip

Facilitating Practice Levels: Practice levels are designed for students to apply their knowledge from the previous levels and develop fluency in using the new blocks of code to solve problems. Students can choose which practice levels they would like to complete, and it's not necessary for a student to complete each practice level before continuing.

Students tend to be more engaged and respond better when they have an authentic choice about how to continue their learning. Allow students to choose practice levels according to their interests and level of comfort, and consider providing opportunities for students to demonstrate and explain their solutions to the practice levels they chose to the entire class.

Digging Deeper: For more tips about programming levels, see the **CSD Guide to Programming Levels**. This document includes strategies and best-practices for facilitating programming levels with students.

7-8 Assessment

7 8

Assessment Opportunity

Formative Assessment: Levels 7 & 8 can be used as a formative assessment.

A rubric is provided in level 7, and written feedback can be given to students. **Click here to learn more about giving feedback to students**.

Level 8 is a free response reflection that can be used as an opportunity to gain insight into how the students completed the assessment level and the choices they made with the HTML tags. In addition, this reflection can be used to identify any misconceptions shared among students that should be cleared up.

Teaching Tip

Facilitating Challenge Levels: Challenge levels are designed as extensions to the concepts and skills students learn throughout a lesson. Challenge levels tend to focus on more open-ended tasks for students to complete, or opportunities to combine several skills from previous lessons together into one program.

Challenge levels do **not** need to be completed for students to meet the core objectives of a lesson. Instead, every task in a challenge level is meant to supplement and enrich the learning objectives of a lesson, but are not required for future lessons. Students can still demonstrate mastery of the objectives of a lesson without completing any of the challenge levels.

Digging Deeper: For more tips about programming levels, see the **CSD Guide to Programming Levels**. This document includes strategies and best-practices for facilitating programming levels with students.

(Optional) Video: One of the challenge levels invites students to create their own poetry as a webpage, featuring an example poem from Caia Lomeli. Caia is a poet and computer science student who was featured in our Poem Art Hour of Code activity. The lesson **links to a video** of Caia discussing how computer science and poetry are similar, especially "starting from a blank page". Even though it's not directly related to webpages and HTML, **you may decide to show this video to students as an inspirational video** on being creative with code, which ties into the projects students will complete in this unit.

Wrap Up (5 minutes)

Reflection

Question of the Day: How can we tell the computer both *what* to put on the web page, and *how* to organize it?

Journal Prompt: In your own words, how does HTML help solve the problem of telling a computer what a web page looks like, not just what content is on it?

Assessment Opportunity

Goal: Students' answers will vary but will likely center around the fact that using tags helps the computer know what different pieces of content "are". Using these tags helps the computer know what the tags are supposed to look like. If this discussion needs to be returned to after students have seen more tags that's fine as well. In either case, use this discussion to motivate the content vs. structure wrap-up point.

As students discuss HTML as a solution, make sure that they are using the key vocabulary of the lesson:

- **website content** - the text and images on a website
- **website structure** - how the content of a website is organized
- **HTML** - Hypertext Markup Language, a language used to create web pages
- **HTML Element** - A piece of a website, marked by a start tag and often closed with an end tag
- **HTML Tag** - The special set of characters that indicates the start and end of an HTML element and that element's type

The content is the literal words that are being typed on the page. Using HTML, students are providing structure to the page, explaining how those pieces of content should be interpreted. Later in the unit students will learn CSS, a language that allows them to individually style elements. For now, however, the styles being applied based on their HTML tags are just the default styles of their web browser. Students don't need to fully understand this difference at this point, as it will be much clearer once they learn CSS later in the unit.

Discuss: After students have had time to reflect individually in the journal, allow them to discuss with a partner, then share with the class.

Remarks

> HTML uses tags to help the computer know what different pieces of content in the web page actually are. Right now we've only learned how to tell the computer that some text is a paragraph, or that part of your website is the body. We've already seen how that affects the way our web pages look and are structured. As we move forward we're going to learn more tags and see more examples of how this language helps us add structure to our webpages.

Review: Return to the list of lesson resources you wrote on the board and review as a class how they are supposed to be used. Refer to the teaching tip above for recommended uses.

9 **Challenge**