Lesson 8: Clean Code and Debugging

Web Lab

Overview

Students deal with common issues that arise when designing web pages in HTML. Students will correct errors in a sequence of increasingly complex web pages. In the process they will learn the importance of comments, whitespace, and indentation as tools for making web pages easier to read. At the end of the lesson students create a list of strategies for debugging web pages and ensuring they are easy to read and maintain.

Purpose

Bugs in HTML are more forgiving than programming languages such as JavaScript (the language used in Unit 3). However debugging is an explicit problem solving process that students will use repeatedly when working with any language on the computer. When problem solving there are different strategies that a computer scientist can use to find the source of the issue.

In addition to the strategies to fix bugs once they have occurred there are certain styles of writing HTML code that help prevent bugs or make it easier to find bugs. The three main style conventions used are comment, whitespace, and indentation. To motivate students to consider using these conventions in the future, the debugging levels demonstrate that it is easier to debug a program that is written with these style conventions.

Agenda

Warm Up (10 minutes)

- **Previous Experience with Bugs**

Activity (35 minutes)

- **Web Lab: Smash Those Bugs!**

Wrap Up (10 minutes)

- **Coding Style Conventions**

View on Code Studio

Objectives

Students will be able to:

- Describe why using whitespace, indentation, and comments makes your code easier to maintain
- Develop a set of techniques for preventing bugs in HTML code and finding them when they occur

Preparation

- [ ] Prepare poster paper, sticky notes, and markers

Vocabulary

- **Bug** - Part of a program that does not work correctly.
- **Comment** - A note in the source code of a computer program that helps explain the code to people who read it
- **Debugging** - Finding and fixing problems in an algorithm or program.
- **Indentation** - The placement of text farther to the right or left of the surrounding text, making it easier to understand the program's structure
- **Whitespace** - Any character that shows up as a blank space on the screen, such as a space, a tab, or a new line; helps separate different parts of the document to make it easier to read

Introduced Code

- `<!-- -->`

Teaching Guide

Warm Up (10 minutes)

Previous Experience with Bugs

Set up: Put a poster up on the wall where all students can get to it. Write the title "Class Bugs" at the top but wait to explain the meaning of the term until it is introduced below.

Group: Place students in groups of 3-4.

Distribute: Give each group a handful of sticky notes

Prompt: With your group:

- Come up with at least three specific problems you have encountered while trying to write web pages in HTML.
- What project were you working on?
- How did you ultimately track down and fix the problem?

Discuss: Have students share out the bugs they have faced and strategies they have used for finding them.

Vocabulary: Introduce the concept of bug and debugging explaining them in the context of the problems and strategies students shared.

Share: Ask each group to use the sticky notes they were given to write down at least three of the bugs they've encountered and how they solved them, using one sticky per bug. Each sticky should have:

1. A brief description of the bug
2. Steps taken to solve it
3. Name of student who solved it

Once groups have written down their bugs, have them stick them up on the class poster.

Remarks

> The problems you had when your HTML code did not work correctly are bugs. The process of fixing bugs and strategies used to fix them is called debugging. Today we're going to be working on our debugging skills and learning some strategies to keep our code clean to help avoid them.

Discussion Goal

Goal: Students do not need to get all the bugs or the strategies in this first discussion. They will add to this list throughout class.

Bugs students might share could include:

- Closing and opening tag switched
- Not closing a tag
- Not putting quotes around attributes values
- Spelling a tag name wrong
- Not putting a list item inside a type of list element
- Using the wrong header tag
- Wrong file path for an image
- Forgetting the equals sign between attribute name and value
- Closing tags out of order
- Not putting content inside a tag

Strategies students might share for debugging could include:

- Guess and Check
- Taking out sections to see which section is causing problems
- Looking for missing tags
- Figuring out what section of the code has the problem (Making the problem smaller)
- Asking a friend for help
- Thinking about when the code last worked and what you have added since then

Teaching Tip

Using the Bugs Poster: Encourage students to use this bugs poster as a resource through the rest of the unit. Whenever a student successfully squashes a new bug, have them put it on the poster. When students are get stuck, they can check the bug poster for someone who may have encountered the same issue before.

Use Journals: If you like you can have sutdents individually track their bugs on a new page in their journals, just like the "HTML Tags" page that they have been updating.

Activity (35 minutes)

Web Lab: Smash Those Bugs!

Group: Put students in pairs to work on these Code Studio levels.

Code Studio levels

Lesson Overview | **Student Overview**

Debugging 2 | 3 | 4 | 5 | 6 *(click tabs to see student view)*

Formatting HTML Student Overview

Challenge 8 *(click tabs to see student view)*

Wrap Up (10 minutes)

Coding Style Conventions

Prompt: What made it harder or easier to debug the web pages?

Discuss: Have students share out things that made it easier or harder to debug the web pages they encounter.

Remarks

> Code needs to be useful for both people and computers. Code that your computer can run might still be really hard for someone (or even you!) to read and make changes to. From now on it's important that we use these practices to ensure our code is easy to read for people, not just good enough for a computer to use.

Discussion Goal

Goal: Students answers will vary but should hopefully include the following:

- Number of bugs
- Use of comments (text between the characters `<!--` and `-->`)
- Separating things onto separate lines (whitespace)
- Grouping together things that are one idea such as a list (whitespace)
- Indenting elements that are inside other elements

If students don't mention some of these things ask them to compare two sites that have different uses of these elements.

Setting Expectations: Use this discussion to motivate the need for making readable code. Highlight that you will expect them to follow the norms they learned today from now onwards, in particular on their projects for this unit.

Standards Alignment

View full course alignment

CSTA K-12 Computer Science Standards (2017)

▶ **AP** - Algorithms & Programming