Learning Objectives:
• Learn the two purposes of comments in a programming language
• Master the syntax for three different styles of comments

Scroll down for content
The purpose of "comments"

Programs can get complicated, as you've already seen. It is very, very useful to occasionally (or often!) narrate the meaning or intention of your code in simple English.

Whenever some part of your code is not obvious to a fellow programmer, you should add a comment explaining it. Don't add too many unnecessary comments, as it clutters your code and obscures the important ones from being noticed.

Another use for comments is temporarily deactivating certain parts of your code so that you can debug more easily. Using the comment symbols for code simply tells JavaScript to treat them as though they were English text, meaningless to it.
Three syntaxes for comments

There are three different syntaxes for indicating comments, two for JavaScript and one for HTML.

1. The one-line JavaScript comment. Precede any line of code with a double slash ("//") and the rest of that line will be ignored by JavaScript. This is useful for brief remarks and explanations. Note that you can put the double slash in the middle of a line, in which case all code after it on that line will be ignored.

2. The block JavaScript comment. Any code that comes between a "/*" character sequence and a "*/" character sequence will be ignored by JavaScript. This is useful for lengthier explanations.

3. The HTML comment. Any tags or text that come between a "<!--" character sequence and a "-->" character sequence will be ignored by the browser. This is useful to explain a tricky part of your HTML code, or to temporarily "turn off" part of it.

Note that these comment types cannot be intermixed! In particular, the "//" and "/* ... */" types cannot be used in HTML mode, and the "<!-- ... -->" type cannot be used in JavaScript mode.
Examples:

<html>
<head>
    <title>Examples of comments</title>
</head>
<body>
    <h1>Here are some comment examples!</h1>

<!-- This sentence will NOT appear in the browser! -->

<script>
    var height; // user's height in inches
    var weight; // user's weight in pounds
    height = parseInt(prompt("How tall are you, in inches?"));

    if (height > 75) {
        alert("You are very tall!");
    /*
        (The following two alert boxes will NOT display.)
        alert("Do you play basketball?");
        alert("Do you ever bump your head on door frames?");
    */
    }
</script>
</body>
</html>
Sources of error

There are two main classes of errors that programmers inadvertently make in their programs:

1. ____________ errors: malformed places where you didn't conform to the requirements of the language. (e.g., leaving out a banana, or misspelling "alert")

2. ____________ errors: mistakes in the algorithm itself. (e.g., subtracting two numbers when you meant to add them)

You will make both types of errors many times in this course. This does not mean you are a bad person. It means you are a person.
Firebug

Firebug is a very useful plugin to Firefox that we will be using extensively in this course. To get it, go to:

https://getfirebug.com

Click on the red "Install Firebug" button, then click the top "Download" link (underneath the first item in the list, representing the most recent version of Firebug), then click "Download Now". Once the program downloads, press the "Install" button on the popup box to add it to Firefox.

You can tell Firebug is installed via the little bug icon in the upper-right corner. Click on the black-and-white bug to change him to color: this activates him.

Firebug gets its name from the fact that a defect in a computer program is called a "______". The process of identifying and removing defects is called "__________________" a program.

Firebug can help you find syntax errors by highlighting in red places where it noticed a problem. This nearly always help you pinpoint the line your error is on. It sometimes also tells you exactly what the error is.

Firebug can help you diagnose logic errors because you can ______ through your program, line by line, viewing JavaScript's Memory Worksheet as you go. You do this by setting a ________________, which tells JavaScript to pause executing its code at a certain line. Stepping through the code one line at a time, and examining the Memory Worksheet in the right-hand pane, can be of vital help in locating a logic error.