

Lab 4.1

Getting Started With JavaScript

In previous labs, we worked with HTML and saw how to use that language to describe how a web page would look. A web browser would read an HTML file (typically via the network) and visually display the page according to the HTML. In this lab, we'll see how we can use the JavaScript programming language to enhance web pages. We can add JavaScript instructions to an HTML file, and the web browser, in addition to displaying the page as described in HTML, can execute those instructions. You can think of this like putting small programs inside web pages, which enables them to much more than is possible with HTML alone.

Vocabulary

All key vocabulary used in this lab are listed below, with closely related words listed together:

input vs. output

Post-lab Questions

Write your answers after completing the lab, but read them carefully now and keep them in mind during the lab.

1. A programming language is designed to be a human-readable (and -writeable!) way to tell computers what to do. Programming languages allow us to give computers instructions. Most computer scientists would agree that JavaScript is a programming language, but many would not consider HTML one. Discuss one similarity between JavaScript and HTML that suggests both might be programming languages.

Discuss one difference between JavaScript and HTML that suggests only JavaScript is a programming language.

2. Consider the concepts of input and output in the context of everyday devices. The telephone takes input via its dialing buttons (or dial, if you have a rotary phone) and the end of the handset you speak into. Phone output comes from the ringer and the end of the handset you listen to. Consider each the following everyday devices and describe their input and output as thoroughly as possible: television, portable CD player.

3. Consider the short JavaScript program you work with in Parts 2 and 3 of this lab. Does this program have input, output, both?

Describe the input and/or output.

How about the currency converter program? Does this program have input, output, or both?

Discussion and Procedure

Part 1. Introduction to JavaScript

Before we dive into working with JavaScript, we'll introduce you to the language and answer a few common questions about it. This background information should help you make sense of the steps required to write and run JavaScript programs.

What is JavaScript? JavaScript is a programming language, and like any programming language, it's used by humans to communicate instructions to computers. JavaScript's most common application is for enhancing web pages. In fact, although JavaScript is in fact used in contexts other than web pages, in these labs, we'll only work with JavaScript with web pages. More specifically, we will never see a JavaScript program by itself; instead, every JavaScript program we write will be written as part of a web page, and we'll use a web browser to run the programs. In this sense, you can think of JavaScript as an extension to HTML. As you'll see in this lab, you literally write JavaScript programs inside your HTML web page files.

Is JavaScript the same as Java? "No," is the short answer. Although the names of the languages are misleadingly similar, JavaScript and Java are in fact very different languages with only superficial similarities. Unlike JavaScript, which (as mentioned above) is generally used to enhance web pages, Java is a general-purpose programming language and can be used for much more than just web programming. (There are e-mail programs, web browsers, web servers, and lots of other applications written in Java, for instance. JavaScript is not well suited for this kind of larger-scale programming.) If you are interested in learning more complicated, more general-purpose programming languages like Java and C++, however, you'll be encouraged to know that JavaScript was deliberately designed to at least superficially resemble these languages, if only in terms of *syntax*. (We first saw "syntax" back in Lab 1.3 when discussing HTML. Loosely speaking, the syntax of a programming language is like the spelling rules, grammar, and vocabulary of the language.)

How do I run a JavaScript program? When a JavaScript program is written in a web page, all you need to do is open the page in a web browser to run the program. All recent versions of popular web browsers (e.g., Mozilla, Netscape, Opera, Internet Explorer) are capable of running JavaScript programs in web pages.

What happened to compilation? Chapter 9 discussed what's happening behind the scenes (or rather, "inside the box") when a computer runs a program. With programming languages like C, C++, and Java, code is *compiled* and translated into machine instructions that the computer can more readily understand and saved in a program file that can then be run by itself. There is special software called the *compiler* which does this translation and saving. In the case of JavaScript, there is no compiler. Instead, web browsers include special software called an *interpreter*, whose job it is to translate the JavaScript code into machine instructions "on the fly," each time the program is run. From the programmer's point of view, this is easier, because they don't need to run the compiler whenever they want to try running their program. On the other hand, since the code has to be translated into machine instructions as the program runs, programs written in languages like JavaScript are often slower than those written in compiled languages.

Where have I seen JavaScript already, perhaps without recognizing it? Many popular web sites use JavaScript in subtle ways in an effort to make their pages more interesting and easier to use. You can use JavaScript to set up your web page so that special effects happen when you click or move the mouse over certain parts of the page. If you've ever filled out a form on a web page or seen images change when you hover your mouse over them, you were probably seeing JavaScript in action.

How do I write a JavaScript program? From reading Chapter 17, you've already seen what JavaScript code looks like, but you might be wondering where you should be putting this code to run it. As we'll see later in this lab, all you need to do is put your JavaScript code in a special section of an HTML file (marked off by tags), and opening that web page in a browser will run the code.

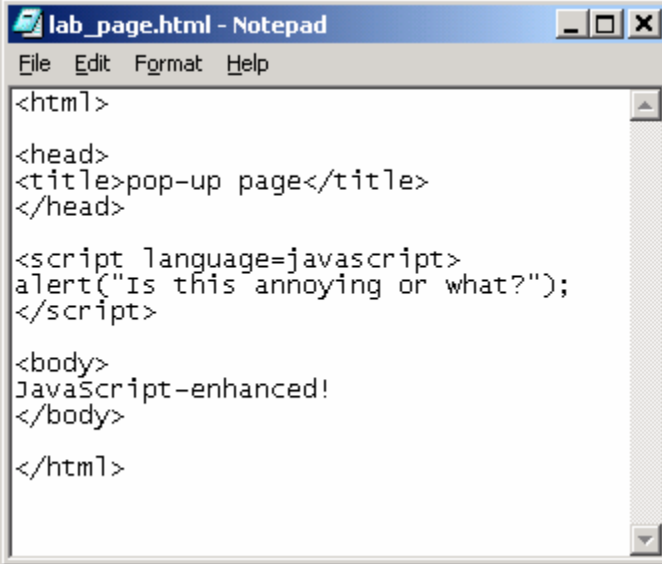
Do I need my own web space or a web server to write and run my own JavaScript? You might be surprised to know that you do not need space on a web server to work with JavaScript. In the same way that you can write an HTML file and open it on a computer that's completely disconnected from the Internet, you can write and run JavaScript programs on a non-networked computer. However, if you want other people to be able to view your JavaScript-enhanced web pages, then you do, of course, need to upload those web pages to a web server somewhere.

What about VBScript? Microsoft's VBScript is another scripting language, similar to JavaScript in many ways, that can be used to enhance web pages. It has the significant limitation that it only works with Microsoft's Internet Explorer, unlike JavaScript, which every popular browser continues to support. This is one of the main reasons why these labs use JavaScript.

Part 2. Writing and running a JavaScript program

The first thing you need to do to start experimenting with JavaScript is to set up our tools: the software we'll use to write and execute the programs. The two main tools we need are a text editor and a web browser. In this lab, we'll assume you're using two commonly available tools on the Windows platform: Notepad and Internet Explorer (IE). However, almost all of these instructions are general enough that you should be able to complete this lab with any text editor and modern web browser.

1. *Run Notepad.* On most Windows PCs, you can find Notepad under the Start menu at **Start \ All Programs \ Accessories \ Notepad**. You can also run it by selecting **Start \ Run...** and entering "notepad".
2. *Type the text shown below into the Notepad window.* This is just an example web page with a very simple JavaScript program (just one line long, in fact!) inside it. The program just pops up a dialog box with a simple message when load the web page in a browser.



```
<html>

<head>
<title>pop-up page</title>
</head>

<script language=javascript>
alert("Is this annoying or what?");
</script>

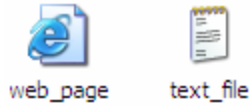
<body>
JavaScript-enhanced!
</body>

</html>
```

3. *Save the file with the .html extension.* You can just name it `lab_page.html` for now.

TROUBLESHOOTING: The inclusion of the `.html` extension is very important. Unless you tell it to do otherwise, Notepad will save your file with the standard `.txt` extension for text files. When you open a file (e.g., by double-clicking it), Windows checks the filename extension to determine what program it should use to open it. If you accidentally save your file with a `.txt` extension, it will open using Notepad instead of your web browser.

Windows will show you what kind of file (plain text or web page) it thinks your file is by showing you one of the following icons:



The notebook icon on the right indicates that it thinks `text_file` is a text file. (The full name of this file, including the extension, is `text_file.txt`.) The file on the right has an Internet Explorer page icon, which indicates the filename has a `.html` extension.

Make sure to leave the Notepad window open when you go on to the next step. We'll be returning to this HTML file to edit the JavaScript program, so it's convenient to leave it open. You can minimize the Notepad window if it gets in the way.

4. *Open the web page you just saved to run the JavaScript program.* If you properly saved the file with the `.html` extension, you should be able to just double-click the file. See the box below for another technique for opening files that some people find faster or more convenient.

What happens when you open the file? Make sure to describe what is displayed in the browser window.

Looking at the web page source (e.g., in your Notepad window), what tags are around the part of the HTML that is actually displayed in the browser window?

What tags are around the part of the HTML that forms your JavaScript program?

Where, if anywhere, is the text inside the **head** tags displayed?

Drag and Drop. There's a very convenient trick you can use to open your web page (`.html`) file in the program of your choice—Notepad for editing and your web browser for running. First, make sure you have the folder containing your web page file open. Next, open a new Notepad window, and resize/move it so it doesn't cover up your web page file. Now drag your web page file onto the Notepad window. Notepad should automatically open the file! You can do the same thing with a web browser window. This works with Internet Explorer, Netscape, Mozilla, and Opera, as well as many other applications. This style of opening files is called “drag and drop.”

5. *Reload the page in the browser window.* What, if anything, happens when you reload the page?

Part 3. “Breaking” the program and other experiments

If all went well in the previous part of the lab, you successfully ran a JavaScript program. It’s valuable to know how things work when you have a correct program, but it’s equally (if not more) valuable to know how your browser will react when you open a web page with an incorrect JavaScript program. After all, even the best of us will frequently have to deal with mistakes (“bugs”) we made writing our programs. For this reason, in this part of the lab, we will concentrate on experimenting with our little HTML file, changing the JavaScript (sometimes intentionally putting bugs in it) and seeing what happens when we run it. (You might remember that we did the same kind of experimenting with HTML in an earlier lab.)

We’ll start simple and make some legal changes to the JavaScript program (i.e., we won’t add any bugs yet). This will help us get used to switching between editing and running the JavaScript. At this point, you should still have a Notepad window with the HTML source and a web browser window with the page open.

6. *Change the alert message.* In the Notepad window, change the alert message string to another short phrase of your choice. Make sure the quotation marks are still around the phrase and remember to save your change when you’re done.
7. *Reload your browser window.* Which message shows up in the dialog box that pops up: the old one or your new one?

All you need to do to update and rerun your program is save in the Notepad window and reload in the browser window.

8. *Add a second **alert** to the program and reload the page.* Back in the Notepad window, copy the **alert** line and paste a copy right after the original **alert** line. Give it a second (different) alert message. Make sure there’s a semicolon at the end of each line and save your changes before running the program.

Describe what happens when you reload the web page.

9. *Change the first **alert** line to the line shown below.* This is the first bug we’ll intentionally introduce.

```
alert(hi);
```

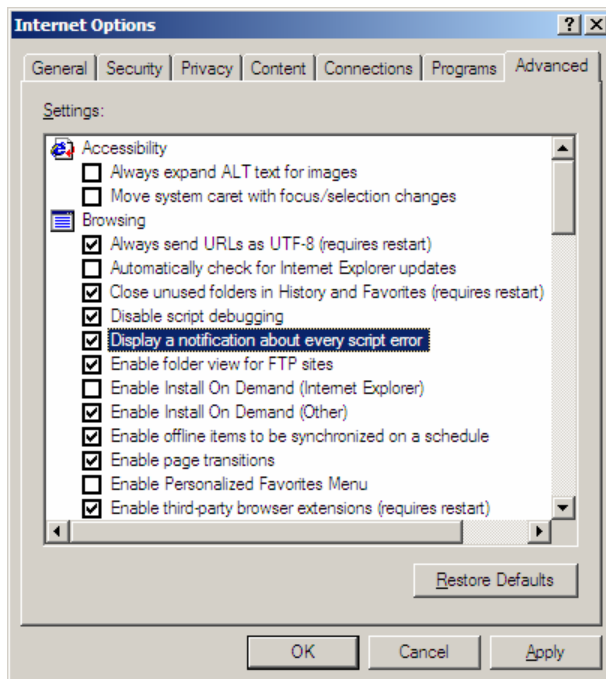
What is the problem with the line above?

10. *Reload the page in your browser.* Describe what happens.

It's evident that something isn't quite working right, because your second **alert** line ought to be fine. We already know what's wrong, but let's use this as an opportunity to learn about some strategies you can use to troubleshoot your JavaScript programs. These strategies should prove helpful later on when you're working with longer, more complicated programs, where it can be more difficult to identify what line(s) of your program are going wrong.

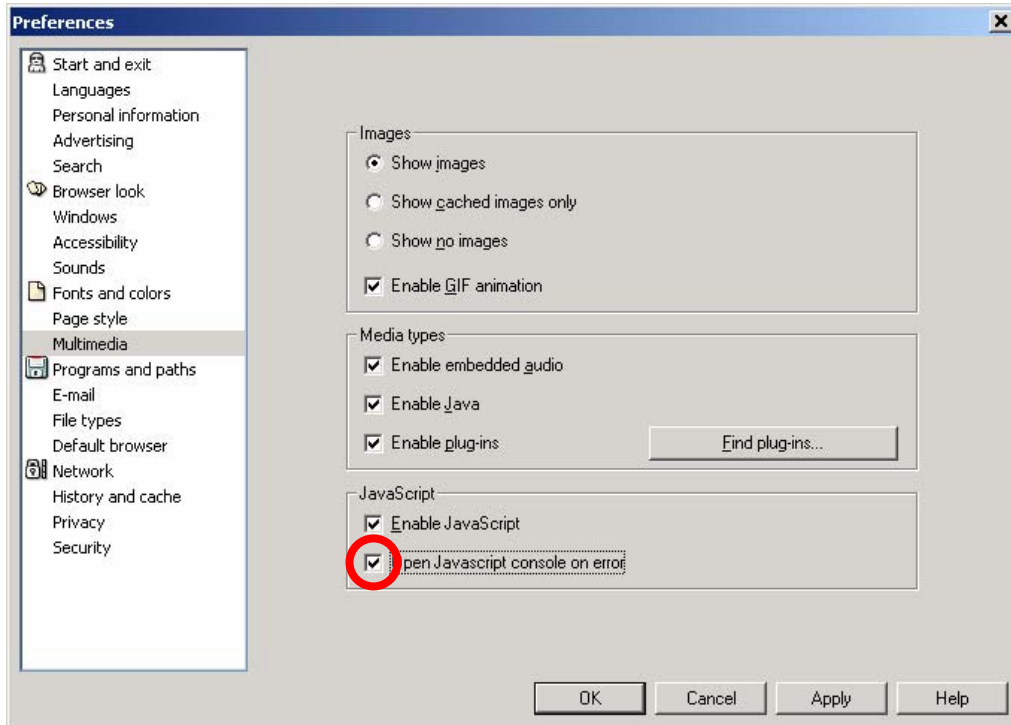
All of the popular web browsers are capable of displaying some information about what went wrong with a JavaScript program, e.g., what line of your script appears to be causing a problem and what kind of problem it's causing. We'll show you how to set up your web browser to display these error messages when a buggy JavaScript is executed.

11. *With Internet Explorer, from the menu bar, select **Tools \ Internet Options...** and click on the **Advanced** tab. Under **Browsing**, find **Display a notification about every script error** and make sure its box is checked.* The next time a JavaScript program causes an error, a dialog box will appear, and clicking the **Show Details** button on the dialog box will allow you to read the error message.

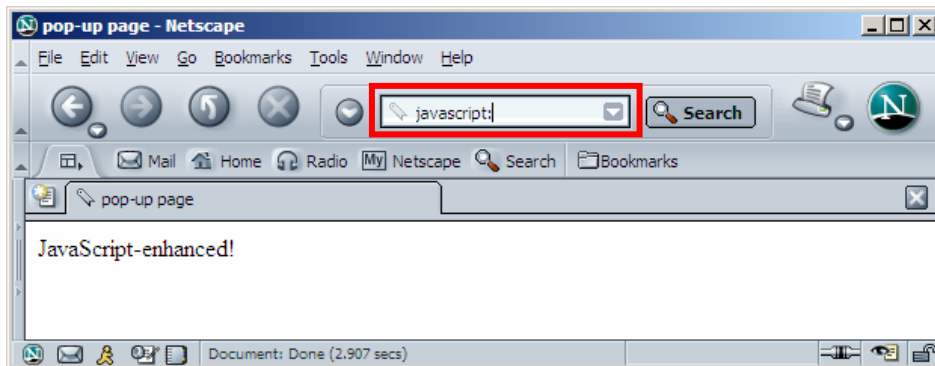


*With Opera, from the menu bar, select **File \ Preferences...** and click **Multimedia** in the list on the left. Make sure the item **Open JavaScript console on error** is*

checked. The next time a JavaScript program causes an error, a separate Opera window where errors are logged will appear.



*With Netscape Navigator or Mozilla, in the navigation toolbar, where you would normally type a URL, enter **javascript:** as shown below. This will open a separate window where any errors from running JavaScript will be logged.*



12. *Try reloading the browser again.* This time, you should get some error message. What line does the browser say the error is on?

TROUBLESHOOTING: Note that Internet Explorer and Netscape/Mozilla number lines starting from the first line of the HTML file, even if the script section starts later in the file. Opera, on the other hand, numbers lines starting from the opening

`<script>` tag, so in the example above, Opera will report a lower line number than what IE or Netscape/Mozilla will.

What is the error, according to the error message?

Indeed, if you hadn't noticed already, a close-parenthesis is missing at the end of the second line of code, just before the semicolon.

13. *Correct the bug and reload the page.* Describe what happens now.

So you want to see line 754 of your HTML file? No, you don't have to move to the top of your file and start counting as you hit the down arrow key! Most text editors have some option to allow you to move to a specific line number in your file. As simplistic as Notepad is, it supports "jumping" to a specific line by selecting, from the menu bar, **Edit \ Go To...** or typing **Ctrl-G** and entering a line number. This is especially handy when a JavaScript error message reports a specific line number in a large file, and you want to find that line in your file to start debugging. This is essential if you're using Internet Explorer, because when it reports a script error, it only tells you the line number and doesn't show you the line of code itself, unlike Netscape/Mozilla and Opera.

Unfortunately, web browsers can only catch certain kind of errors and report them. Experiment with each of the following changes and determine whether an error message results.

14. *"Break" the opening `<script>` tag by removing the `<` character.* Does reloading the page result in an error message? What is displayed on the web page?
15. *Undo the last change and remove the close-parenthesis in the second `alert` line.* What, if any error message, results when the page is reloaded?

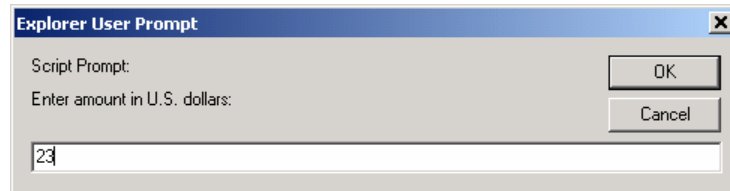
Undo the change to end up with a properly working program before proceeding.

Part 4. Simple Currency Converter

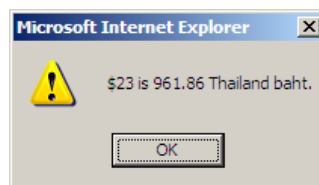
To finish off this lab, you're going to write a simple web page that might be useful for someone about to travel to a foreign country. The page will let a user figure out how much a certain dollar amount is worth in the currency of a foreign country. (You get to choose which country...Which would you like to visit someday?)

You'll need a couple new programming "building blocks" to complete this. You'll probably find it helpful to use a variable. (Review Chapter 17 if you've forgotten how to work with variables.) You'll also find **prompt** useful for getting user input. Similar to **alert**, **prompt** pops up a dialog box, but the **prompt** dialog box allows the user to enter a value. Your program should start by using **prompt** to ask the user for an amount in U.S. dollars, like this:

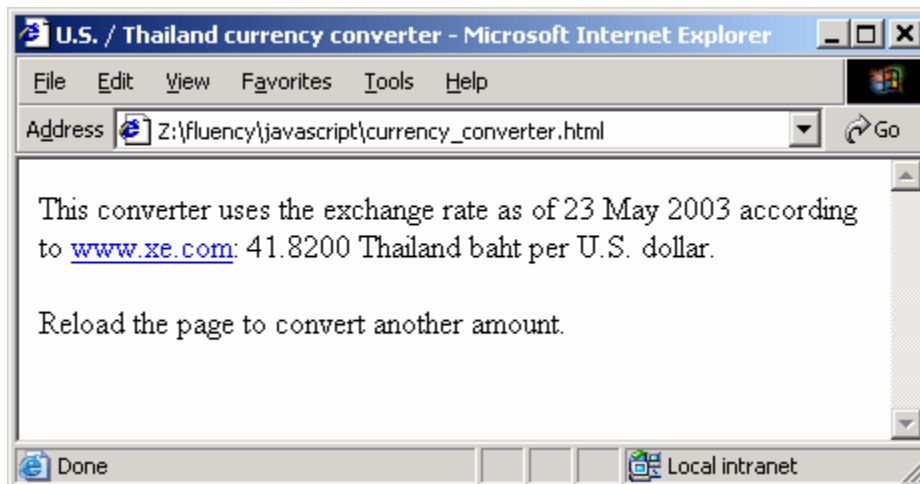
```
var dollarAmount =
  prompt("Enter amount in U.S. dollars:");
```



When the user enters an amount, e.g., **23** like shown above, an **alert** dialog box should display both the entered value and the equivalent amount in a foreign currency. For this example, we've chosen Thailand's currency.



Meanwhile, after this dialog box is OK-ed, the web page should provide some background information and instructions:



“You must BECOME the computer.” Forget which way *input* and *output* go? Remember that these terms are defined from the computer's point of view. If you imagine that you are the computer, input is information that comes *into* the computer

from the user, and output goes *out* of the computer to the user. We commonly say a computer or a program “takes” input and “produces” output, but for short, you can use these words as verbs, too: “The computer/program *inputs* data and *outputs* some computed result.”

You can either use a search engine to find current currency exchange rates on the web (try keywords “exchange rate”), or you can just make up your own currency and pick an exchange rate yourself. (If you were the leader of your own nation, what would your currency be called?)

You should end up with a very short program—several lines at most. If you don’t remember how assignments, mathematical expressions, and how the `+` operator works with strings, you should review Chapter 17 once more.

At this stage of the game, it’s perfectly alright for your program not to work properly if the user enters something that doesn’t make sense for the dollar amount, e.g., a word or nothing at all instead of a number. We’ll learn more about dealing with this kind of situation in a later lab.

Optional Additional Activities

- Try adding some flexibility to your exchange rate page by letting the user input the exchange rate they want to use, via another **prompt** dialog.
- Try moving the `<script>` section of your HTML file below the `<body>` section. Does the page behave differently when loaded in a browser? What happens when you put the `<script>` section somewhere inside the `<body>` section?

Further Reading

- JavaScript is a very popular language for web development, so there are countless web sites dedicated to teaching and sharing JavaScript lessons and tricks. Try checking the major web indexes, like Yahoo and the Open Directory Project:
http://dir.yahoo.com/Computers_and_Internet/Programming_and_Development/Languages/JavaScript/
<http://dmoz.org/Computers/Programming/Languages/JavaScript/>
<http://directory.google.com/Top/Computers/Programming/Languages/JavaScript/>