Contents

[This Course 2](#_Toc484865461)

[Grades 3](#_Toc484865462)

[Submitting Homework 3](#_Toc484865463)

[Contacts 3](#_Toc484865464)

[Introducing JavaScript 4](#_Toc484865465)

[Not Allowed 4](#_Toc484865466)

[Allowed 5](#_Toc484865467)

[Useful Sites 5](#_Toc484865468)

[Our First JavaScript Scripts 5](#_Toc484865469)

[Where to Write JavaScript 7](#_Toc484865470)

[Body JavaScript 8](#_Toc484865471)

[Head JavaScript 9](#_Toc484865472)

[External-File JavaScript 12](#_Toc484865473)

[Defer 12](#_Toc484865474)

[Comments 14](#_Toc484865475)

[Console 16](#_Toc484865476)

[Four Important Objects 18](#_Toc484865477)

[Different DOMs 24](#_Toc484865478)

[Variables 26](#_Toc484865479)

[JSLint 28](#_Toc484865480)

[Data Types 31](#_Toc484865481)

[Lab 1 32](#_Toc484865482)

# This Course

Day 1: Lesson 1

Lab 1 (take it home; submit it online before lesson 2)

Assignment 1 (take it home, submit it online before lesson 6)

Assignment 2 (take it home, submit it online before lesson 6)

Day 2: Quiz 1 (closed book; at the start of class)

Lesson 2

Lab 2 (take it home; submit it online before lesson 3)

Day 3: Quiz 2 (closed book; at the start of class)

Lesson 3

Lab 3 (take it home; submit it online before lesson 4)

Day 4: Quiz 3 (closed book; at the start of class)

Lesson 4

Lab 4 (take it home; submit it online before lesson 5)

Day 5: Quiz 4 (closed book; at the start of class)

Lesson 5

Lab 5 (take it home; submit it online before lesson 6)

Day 6: Quiz 5 (closed book; at the start of class)

Lesson 6

Lab 6 (in class)

Exam (open book; at the end of class)

Assignments due

# Grades

Check your grades on <http://my.bcit.ca>

|  |  |
| --- | --- |
| Quiz 1 | 8% |
| Quiz 2 | 8% |
| Quiz 3 | 8% |
| Quiz 4 | 8% |
| Quiz 5 | 8% |
|  |  |
| Lab 1 | 3% |
| Lab 2 | 3% |
| Lab 3 | 3% |
| Lab 4 | 4% |
| Lab 5 | 4% |
| Lab 6 | 3% |
|  |  |
| Assignment 1 | 10% |
| Assignment 2 | 10% |
|  |  |
| Exam | 20% |

# Submitting Homework

Submit homework to BCIT’s D2L (Desire to Learn) dropbox folder for this course. It is located in our course section of COMP2015 at <http://learn.bcit.ca> (COMP2015 🡪 Activities 🡪 dropbox).

# Contacts

Your instructor can be reached at Jason\_Harrison@bcit.ca or twitter.com/BCITJason

NOTE: BCIT’s mail server strips any JavaScript out of emails and email attachments. If you have questions, put JavaScript without the <script> tags into the body of the email instead.

Each student has had an email account created by BCIT. It can be found at http://my.bcit.ca

Please check that email regularly, or have it autoforwarded to an email account which you do check regularly: E-mail 🡪 Options 🡪 Settings 🡪 Mail Forwarding

# Introducing JavaScript

* JavaScript was originally called LiveScript
* JavaScript has nothing to do with Java
* JavaScript was invented by Netscape in 1995
* JavaScript was the first client-side scripting language
* JavaScript runs in the browser
* There are essentially millions of types of browsers
* Each browser has a different way of running JavaScript
* One of our most difficult challenges is getting our scripts to behave similarly in every browser
* ECMAScript is the standardized version of JavaScript

# Not Allowed

* Read (or write to) the client’s file system
* Cannot hide JavaScript
* Send emails without the client’s knowledge/consent
* Close browser windows it did not open
* Execute programs on the client’s computer
* Open tiny windows (less than 100px x 100px)
* Establish connections to other computers
* Set file-upload inputs
* Read the browser’s history (links, yes; read, no)

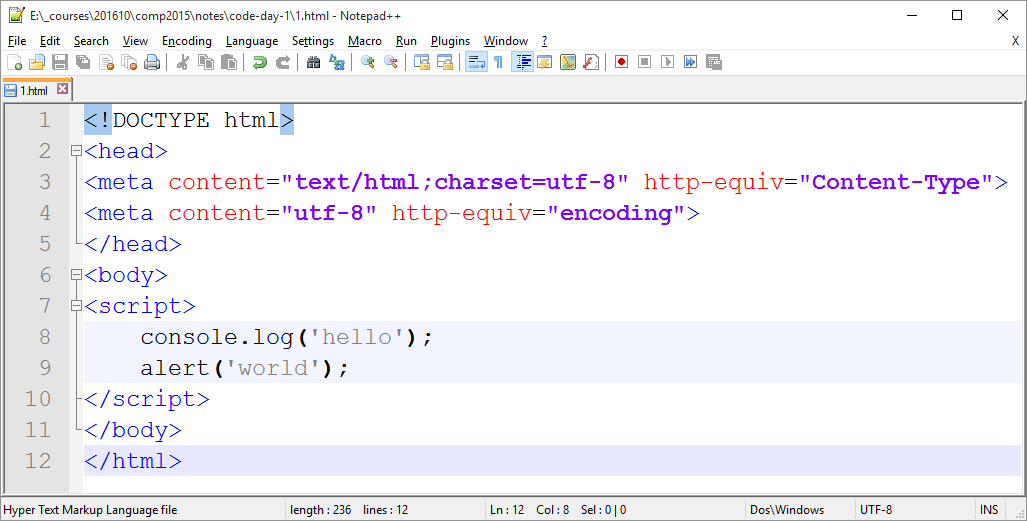
# Allowed

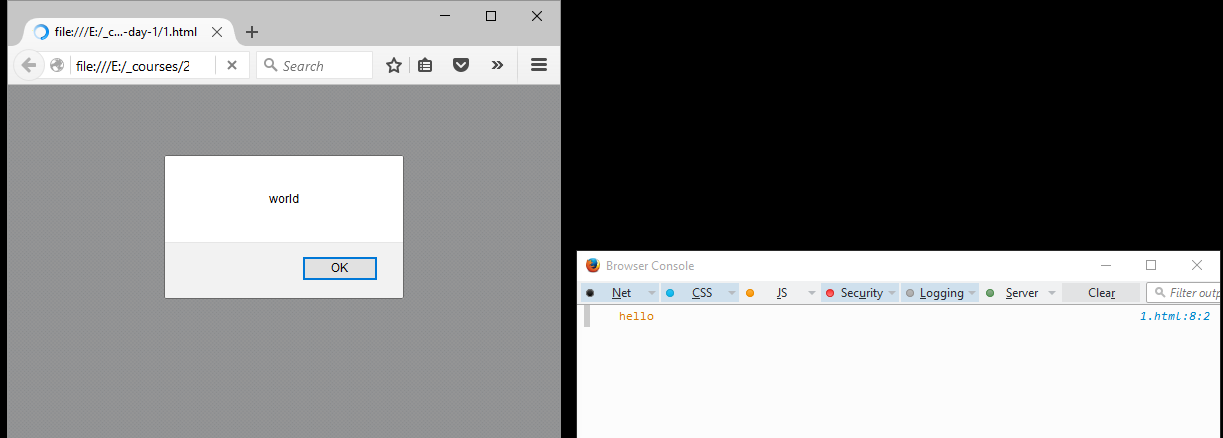
* AJAX
* DOM
* HTML
* CSS
* JSON
* Timers and event handling
* (NodeJS on the server too)

# Useful Sites

|  |  |
| --- | --- |
| Mozilla Developer Network | https://developer.mozilla.org/en-US/docs/Web/JavaScript |
| JavaScript IDE | https://www.jetbrains.com/webstorm |
| ECMA Standards | http://www.ecmascript.org |
| Free JavaScript Book | http://eloquentjavascript.net |
| JavaScript Lint Tool | http://www.eslint.org |
| Help | http://stackoverflow.com |
| Browser add-ons/extensions | http://codecondo.com/top-15-mozilla-firefox-addons-web-developers/ |
| Free stuff | https://education.github.com/pack |

# Our First JavaScript Scripts





In between the <script> and </script> tags, only JavaScript can appear. Do not put HTML.

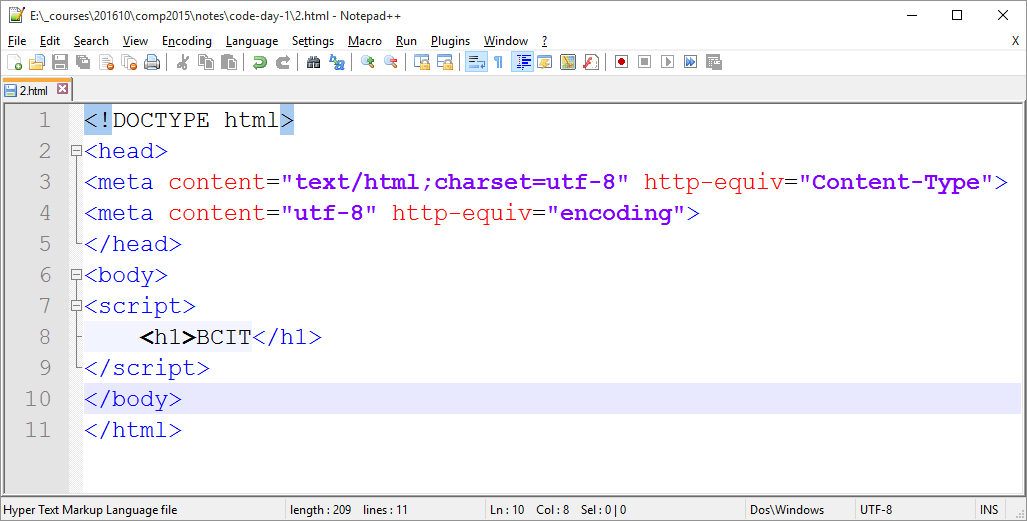


Figure 1: Error

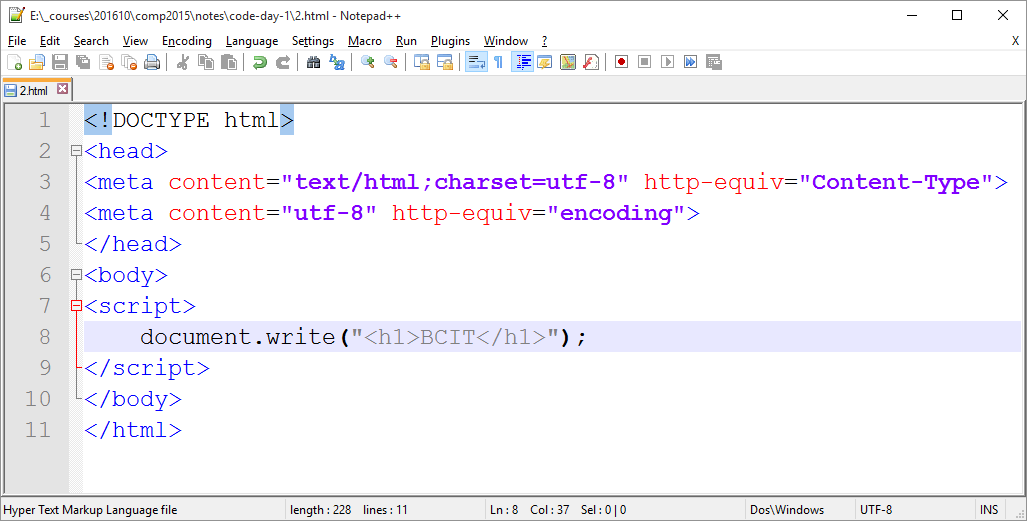
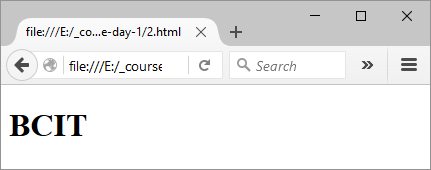


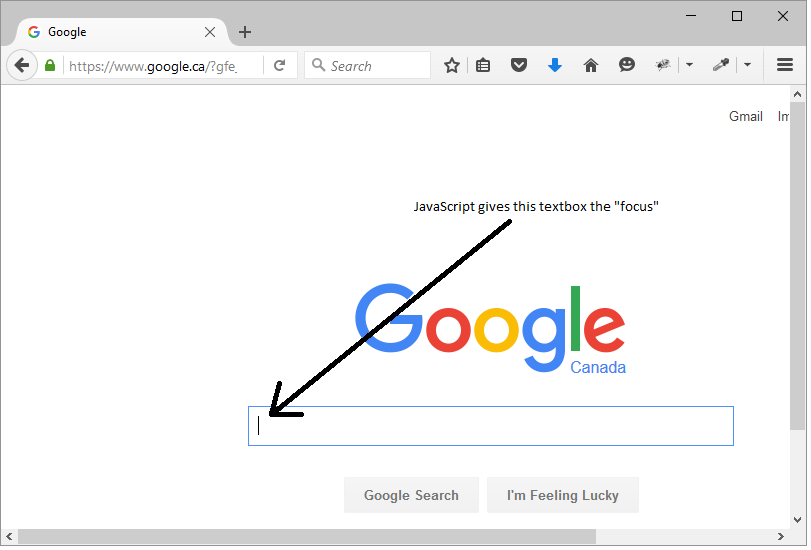
Figure 2: This is OK



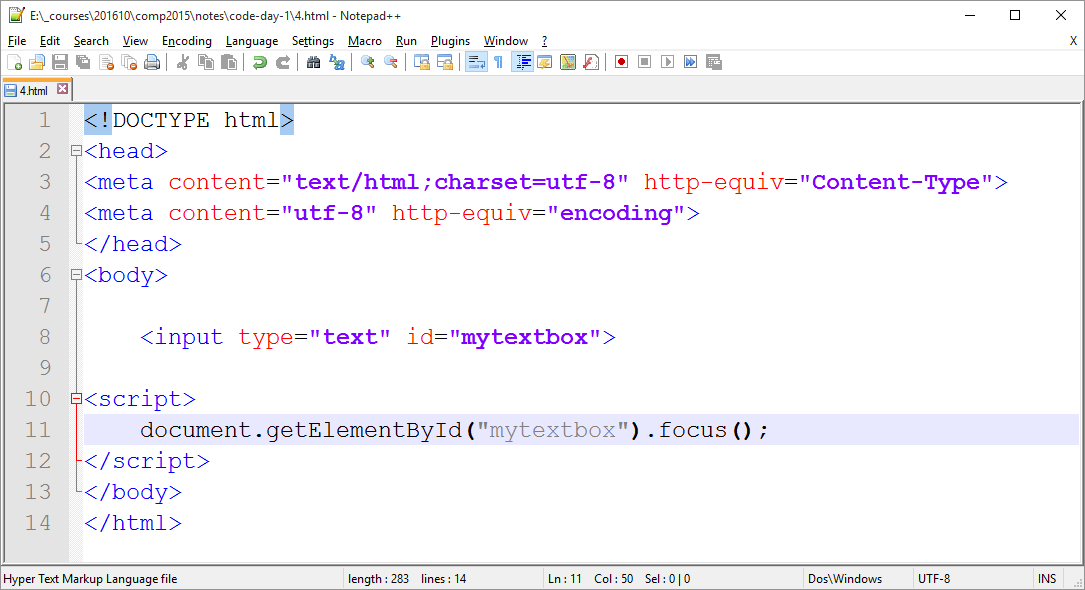
# Where to Write JavaScript

We can write JavaScript in the <head> section, or the <body> section, or in an external <script> section (its own file).

Let’s write a simple script that “automatically” puts the focus onto a text box when the page loads. Google does this:



## Body JavaScript



Don’t pay too much attention for now to the getElementById() instruction. It is very important, but the most important thing right now is that the instruction is *inside the body*.

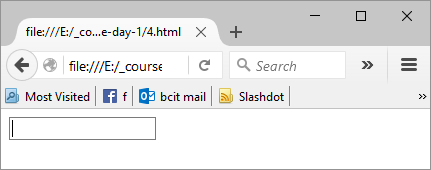
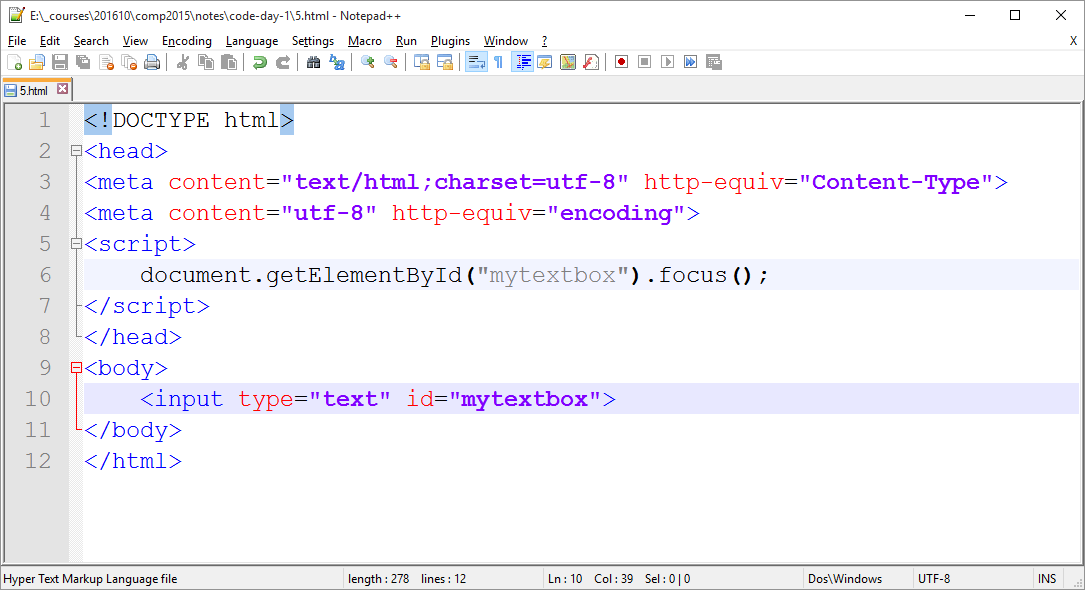


Figure 3: It Worked!

## Head JavaScript



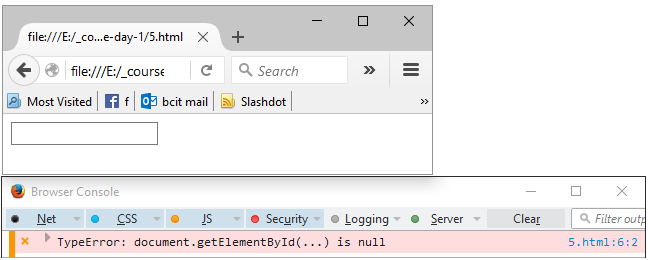


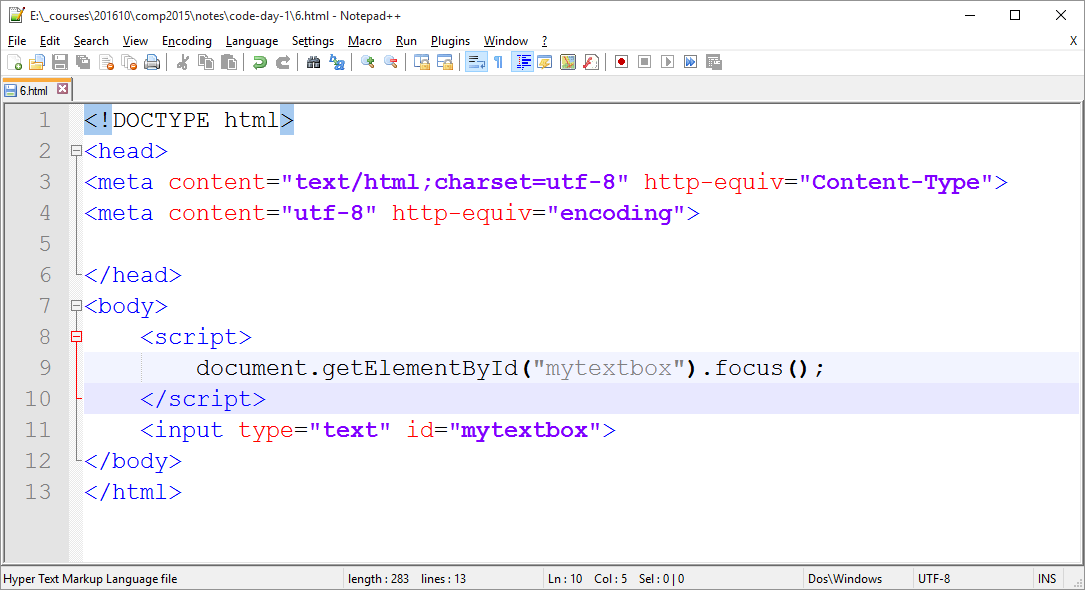
Figure 4: Does NOT Work from the <head>!

In this case, we moved the exact same JavaScript instructions from the body section to the head section, and now the textbox does *not* get the focus.

Although the head section is very often a great place to put JavaScript code, some changes need to be considered first.

The <head> section of a document gets fully loaded before any of the <body> section gets loaded. Because of this, JavaScript running in the <head> cannot find the element whose ID is “mytextbox” because it doesn’t exist yet.

We did not have that problem with our original code because it was loaded in the <body> section…specifically, *after* the textbox was created. The code would have an error if we moved our JavaScript above/before the textbox code, even if it were in the <body>:



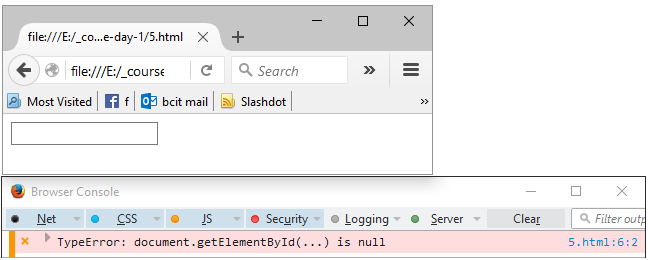
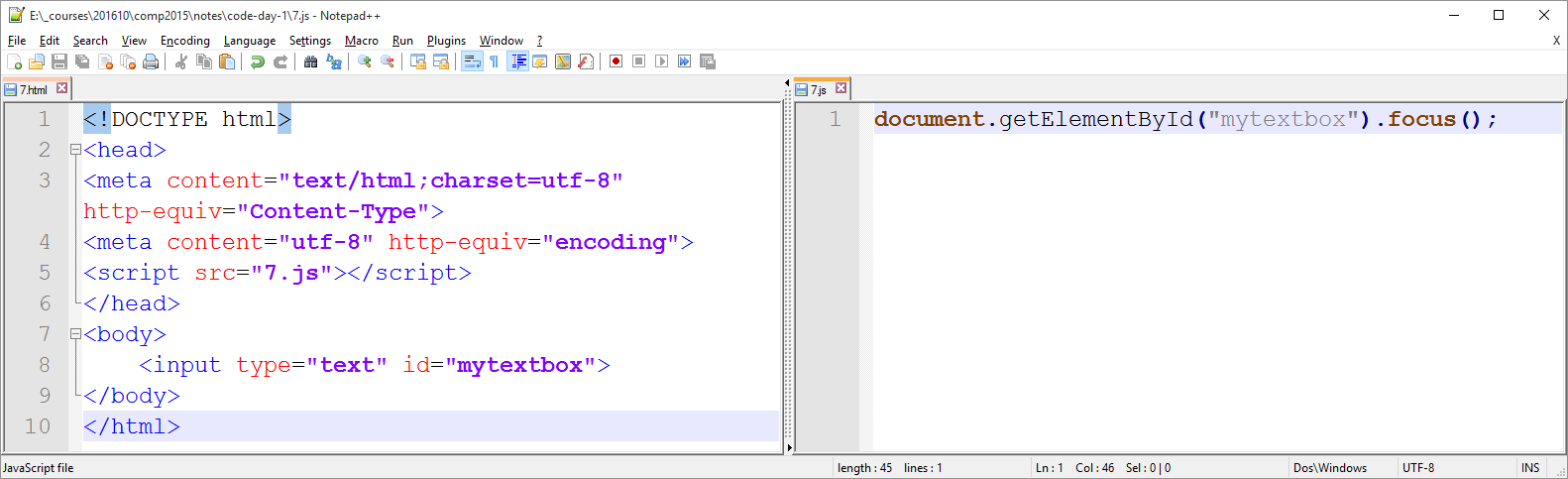


Figure 5: Does Not Work BEFORE the TextBox Exists!

What we really need in the <head> section is a way to “delay” the JavaScript from running until the <body> has been fully loaded. There is a way to do this. But first, let’s put our JavaScript into an external file and run it that way.

## External-File JavaScript



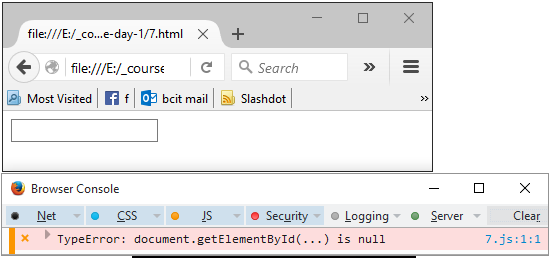
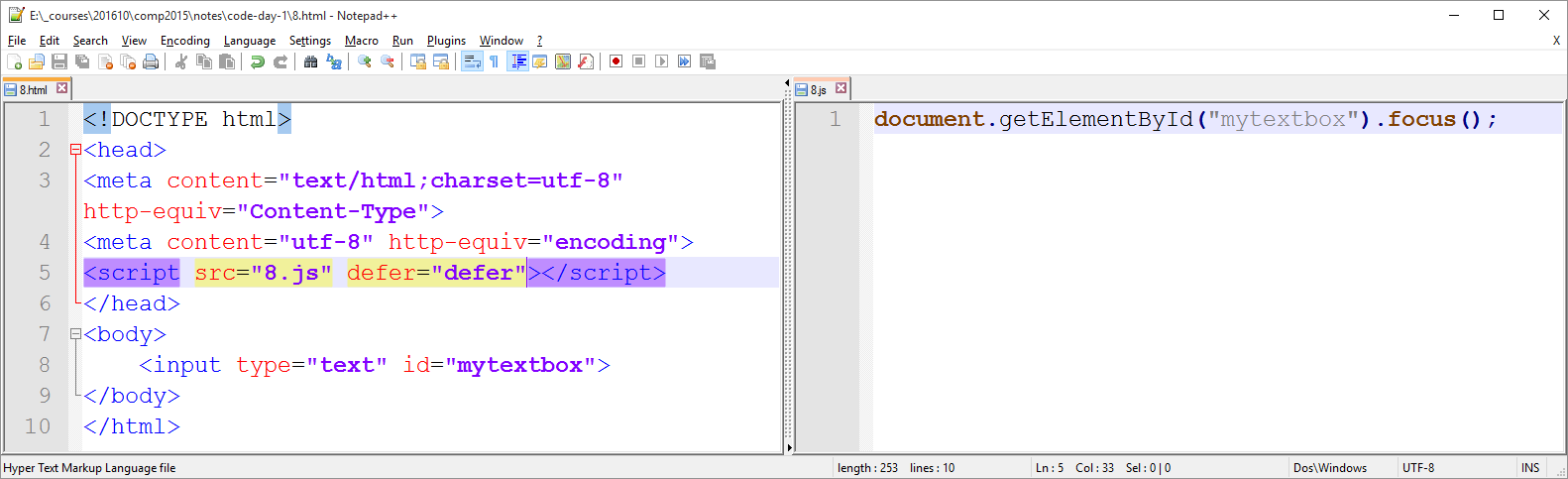


Figure 6: Does Not Work, Again!

## Defer

Again…we need to *delay* the external script from executing. In particular, a deferred external JavaScript will begin downloading right away, but will not execute until the entire document is rendered. After the page is finished, all deferred external JavaScripts will execute in the same order in which they were declared.

NOTE: you can only declare *external* scripts (with the “src=…” attribute).



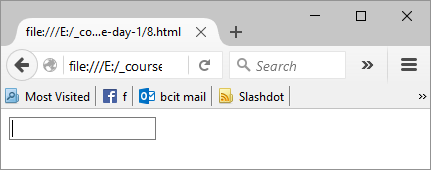
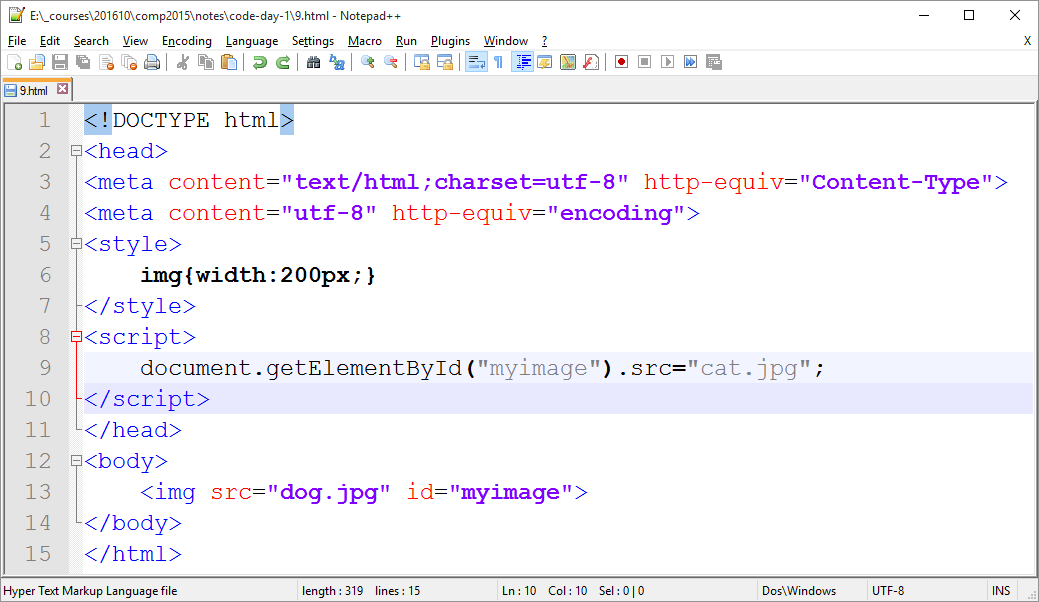


Figure 7: It works! The Focus is Working!

We can write JavaScript which interacts with our web page (our “document”) by:

* Putting our <script> section at the very bottom of the <body>
* Putting our <script src=”…” defer=”defer”> anywhere, even in the <head>

What is wrong with the following code? Why won’t it work? Fix it.



NOTE: There are techniques available to defer *internal* scripts as well, but we will look at these later.

# Comments

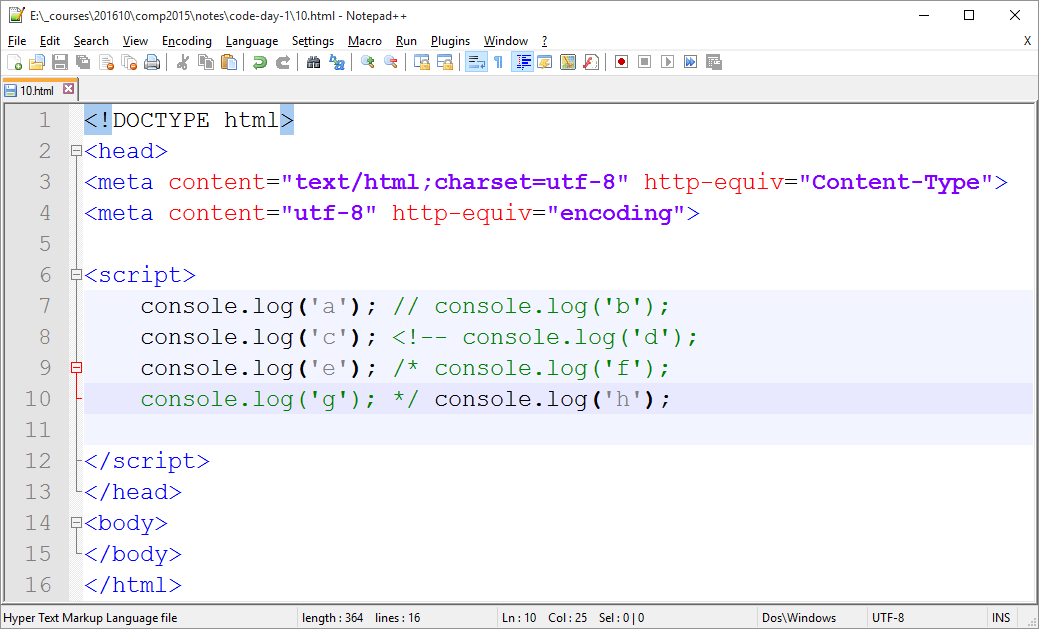
Comments are “notes” that are ignored by JavaScript and are intended for humans to read instead.

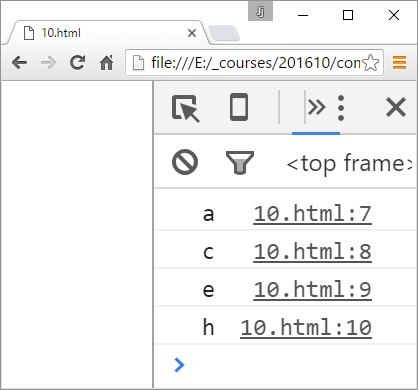
There are three types of comment tags used in JavaScript:

1. // two forward slashes comments out everything to the right of the // until

the end of the line

1. <!-- this acts *exactly* the same as //
2. /\* comments out everything until \*/ is found; this can span multiple lines





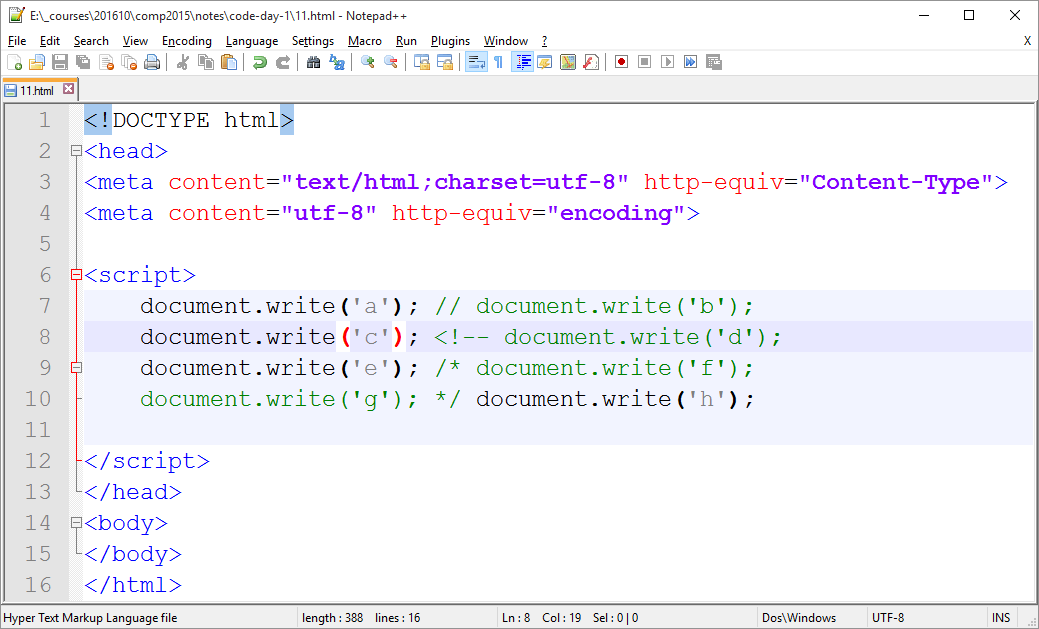
The console shows us that:

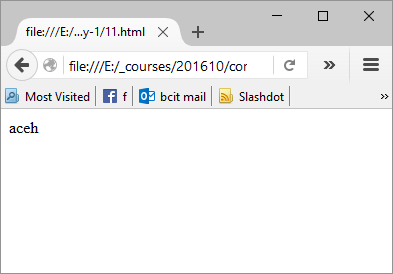
* ‘b’ is commented out by //
* ‘d’ is commented out by <!--
* ‘f’ and ‘g’ are commented out by /\* and \*/

# Console

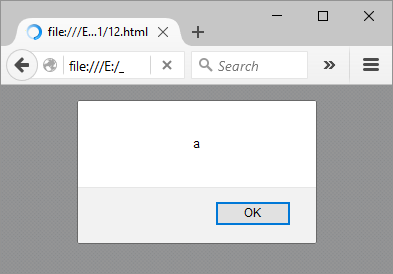
You can get to the console in Firefox or Chrome by pressing Ctrl-Shift-J

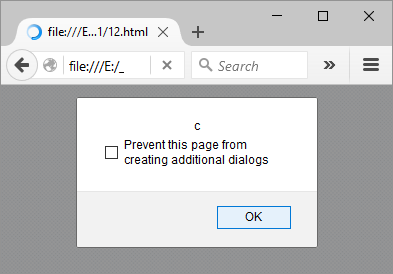
Instead of logging our results to the console, we could write the document:

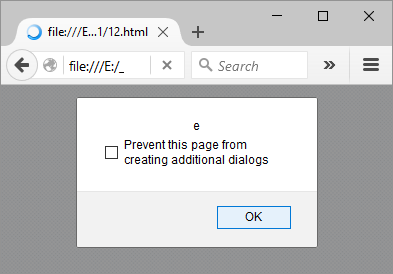


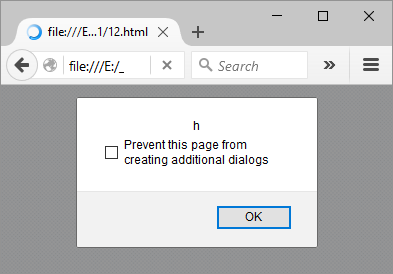


…or we could use the window.alert instruction instead:









# Four Important Objects

There are four important objects with which we want to get familiar:

1. document
2. window
3. style
4. navigator
5. document: the HTML web page itself, from <html> to </html>

example 1: document.getElementById(“mytextbox”)

example 2: document.body.bgColor=”red”

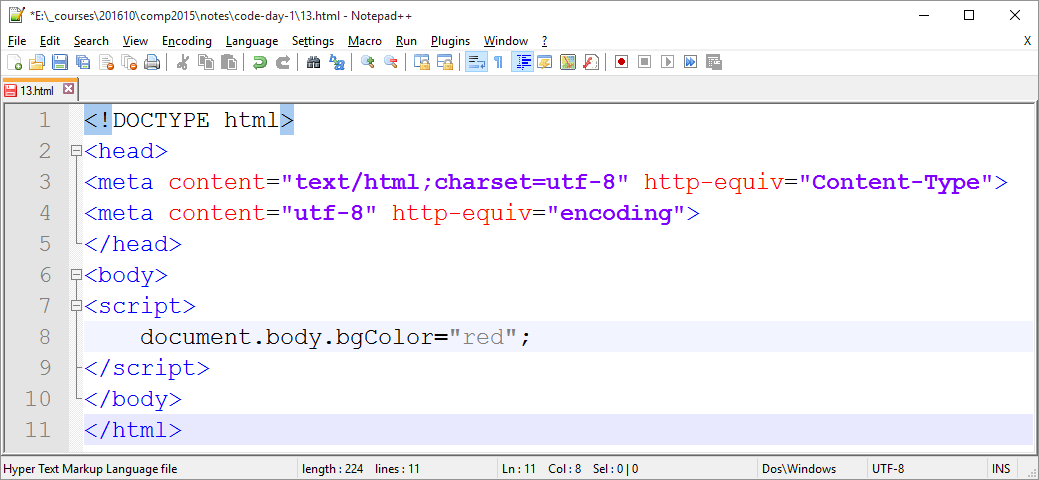
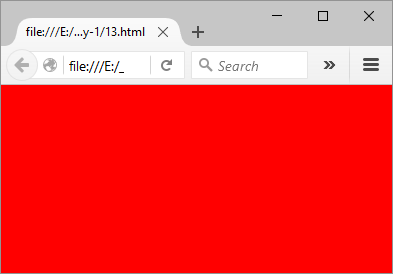


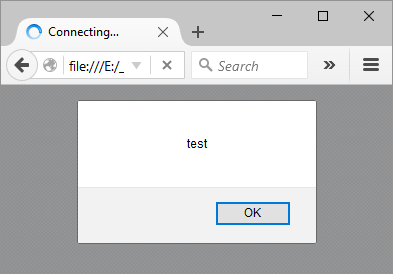
Figure 8: Make Sure this is in the Body Section!



1. window: the current containing window (or tab)

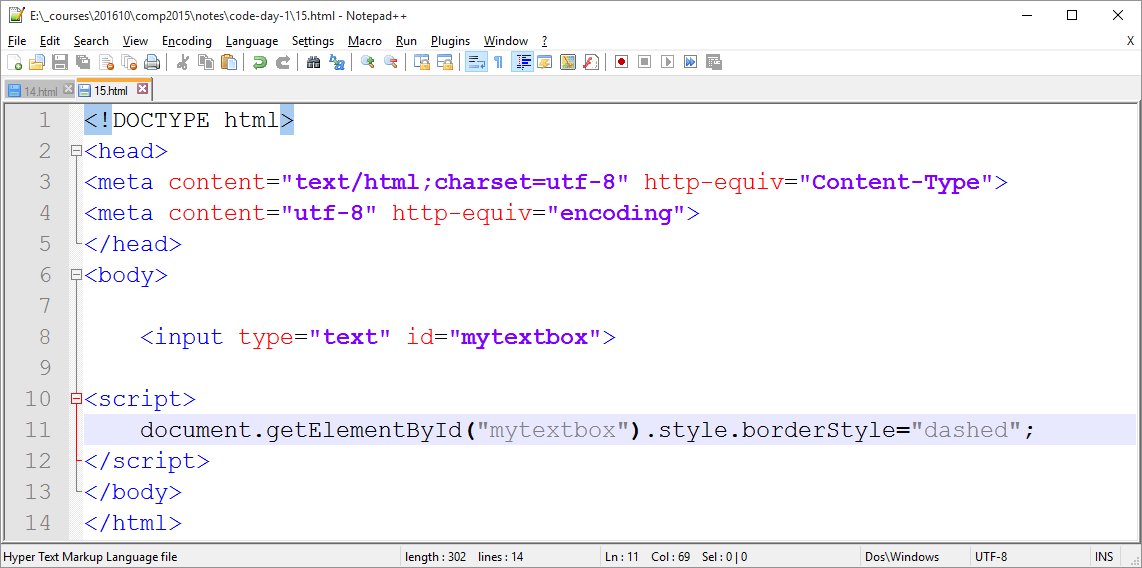
example 1: window.document.body.bgColor=”red”;

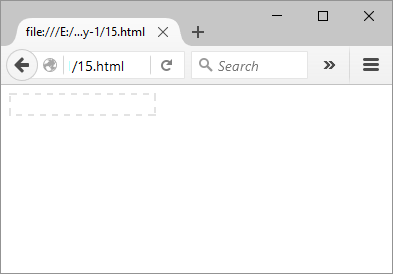
example 2: window.alert(“test”);



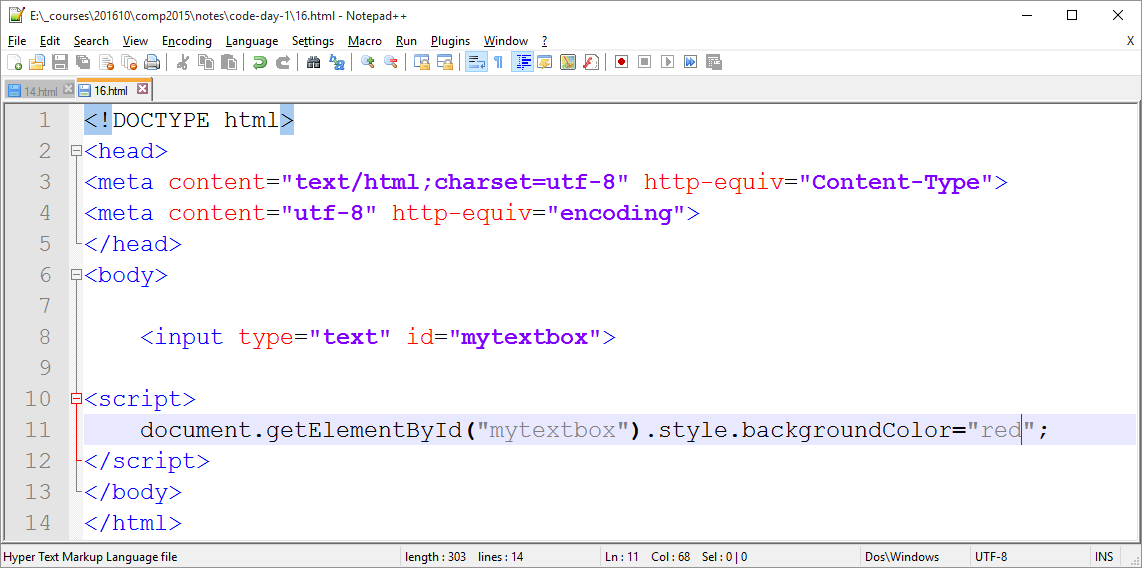
1. style: the CSS properties of an element

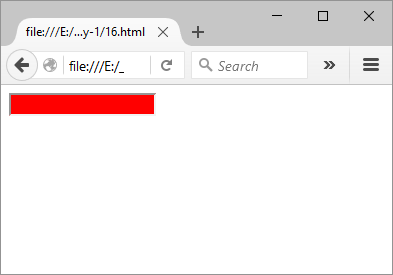
example 1: document.getElementById(“tb”).style.borderStyle=”dashed”





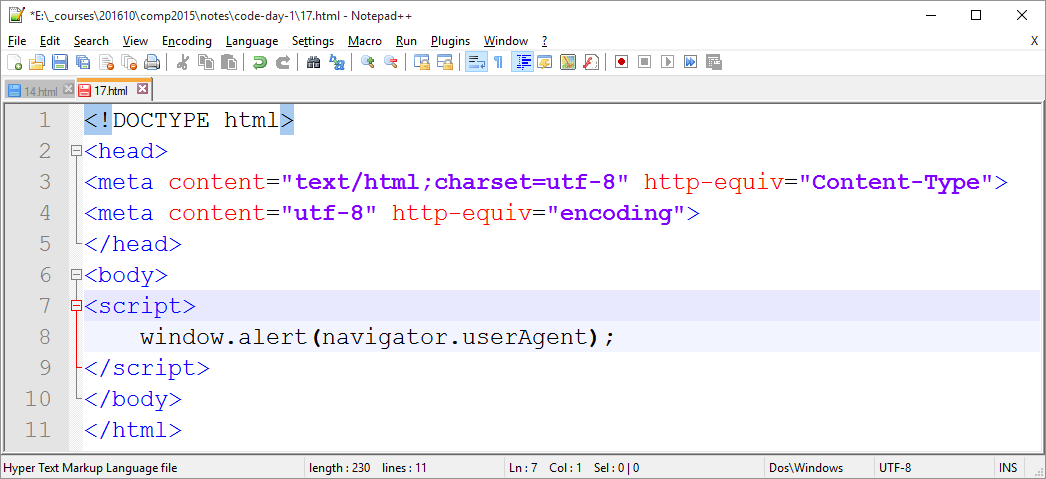
example 2: document.getElementById(“tb”).style.backgroundColor=”red”;

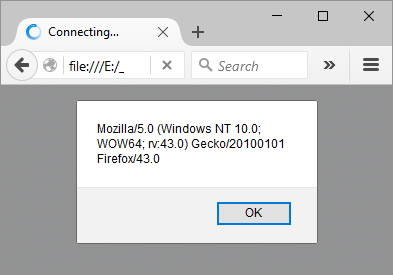




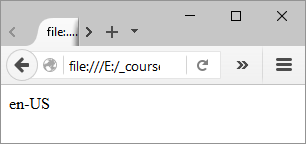
1. navigator: the browser itself

example 1: navigator.userAgent





example 2: navigator.language

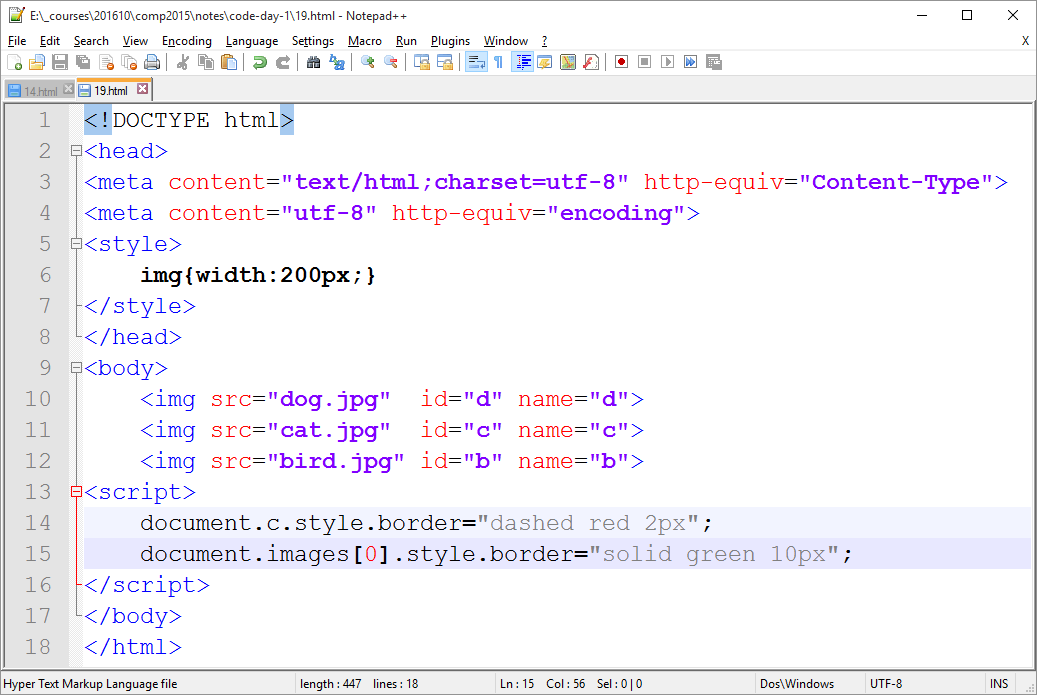


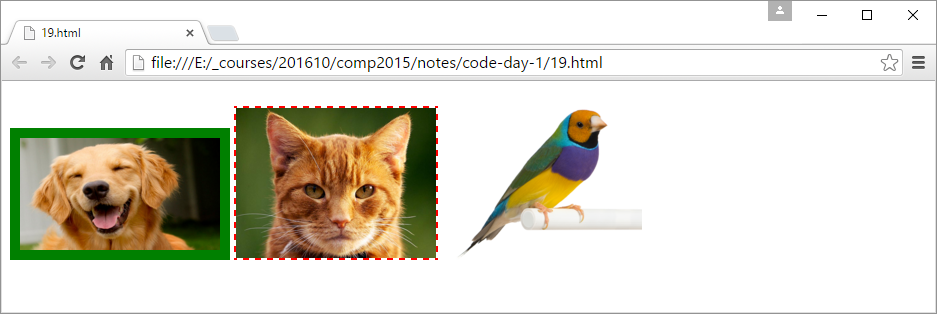
# Different DOMs

The Document Object Model (DOM) is a set of code JavaScript can use to interact with the browser and the Web Page.

Since the mid 1990s, the DOM has evolved. The history in a nutshell is:

* Browsers existed first without any JavaScript
* JavaScript was then implemented in Netscape
* JScript – a JavaScript clone – was then implemented in Internet Explorer
* DOM 0 is now used to refer to these “olden days” in which each browser did things their own way. Most references to the web page’s elements were done either very awkwardly or with respect to the *position* of the element on the page. For example:





DOM 0 is a very convenient way to reference some of a page’s collections:

document.images is a collection of all the images; document.images[0] is the first image on the page, etc….

document.forms is a collection of all the forms; document.forms [4] is the fifth form on the page, etc….

document.links is a collection of all the hyperlinks; document.links [99] is the hundredth link on the page, etc….

There is no DOM 0 shortcut for other collections, such as document.paragraphs, etc…. For those, we will use newer DOMs such as DOM 1.

* Netscape and Internet Explorer battled to create the best browsers and to set the new standards. Internet Explorer used the document.all DOM as follows:

document.all[‘dog’].src = ‘cat.jpg’;

Netscape used the document.layers DOM instead (now extinct).

* Eventually, the W3C came to the rescue and implemented the new DOM standards, in the best interests of web developers and web users alike. The new code looked like:

document.getElementById(“d”).src=”cat.jpg”

and

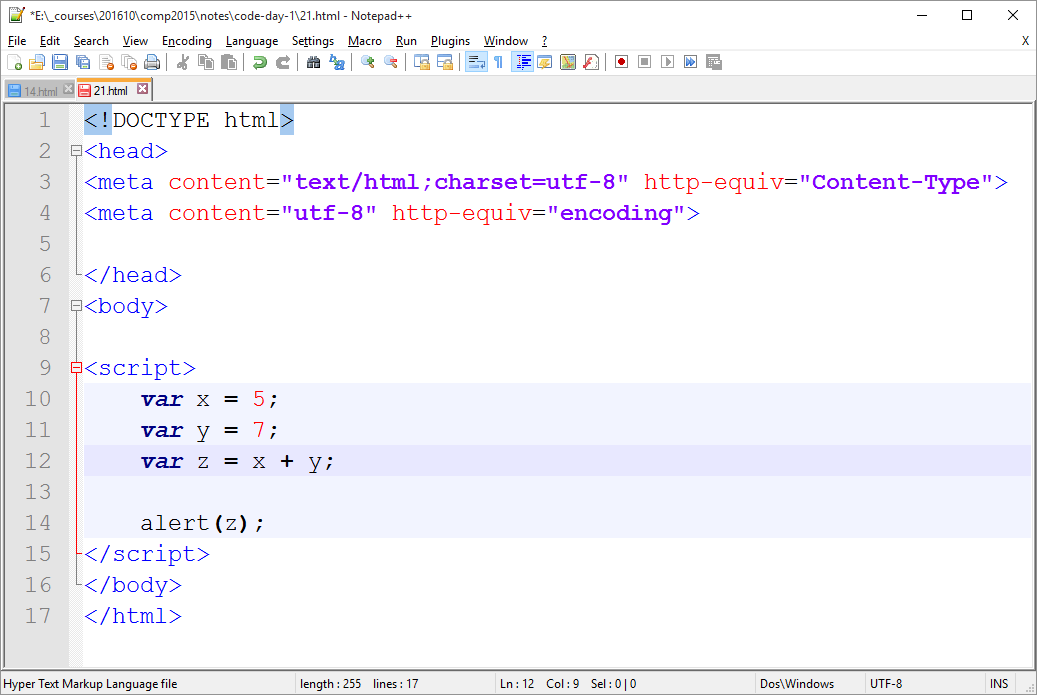
document.getElementsByTagName(“p”)

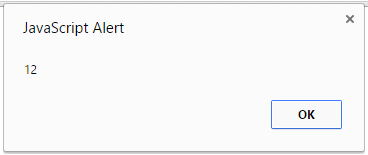
etc….

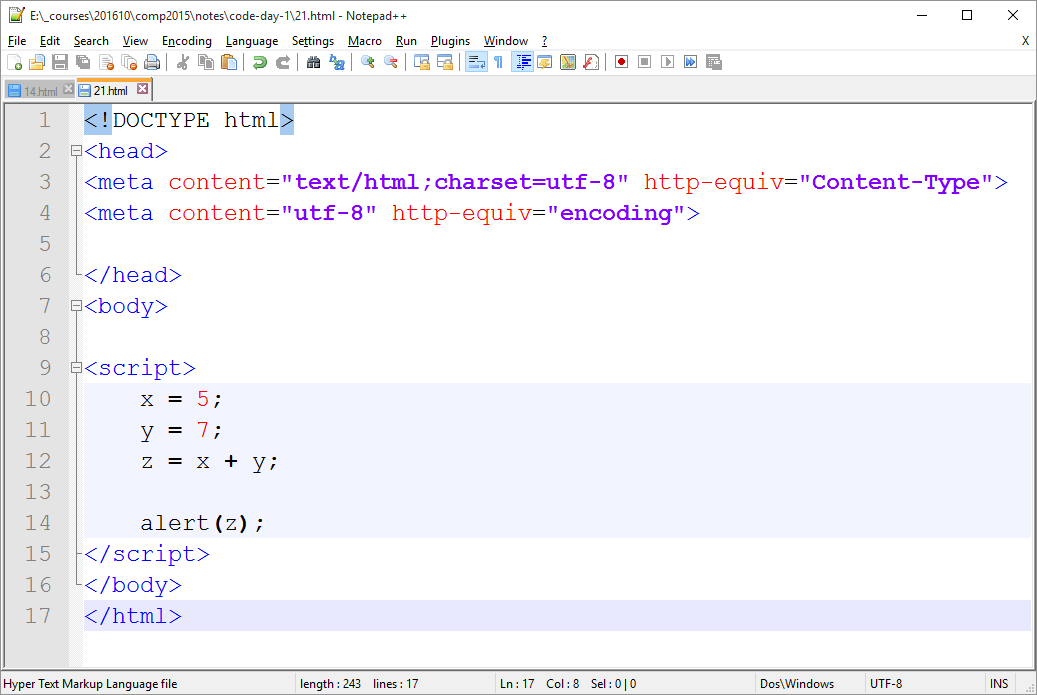
We will use DOM 1 (and 2 and 3; also created by the W3C) until we later augment it by using jQuery.

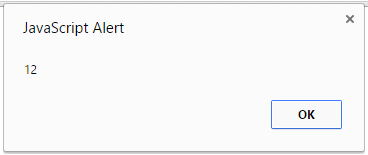
# Variables

We can create variables of different data types. We can do this with or without using the keyword “var”. It is almost always expected that you would use the keyword var; otherwise your variable becomes essentially a global variable, which is not good.





Note that the following code seems to work the same way: 

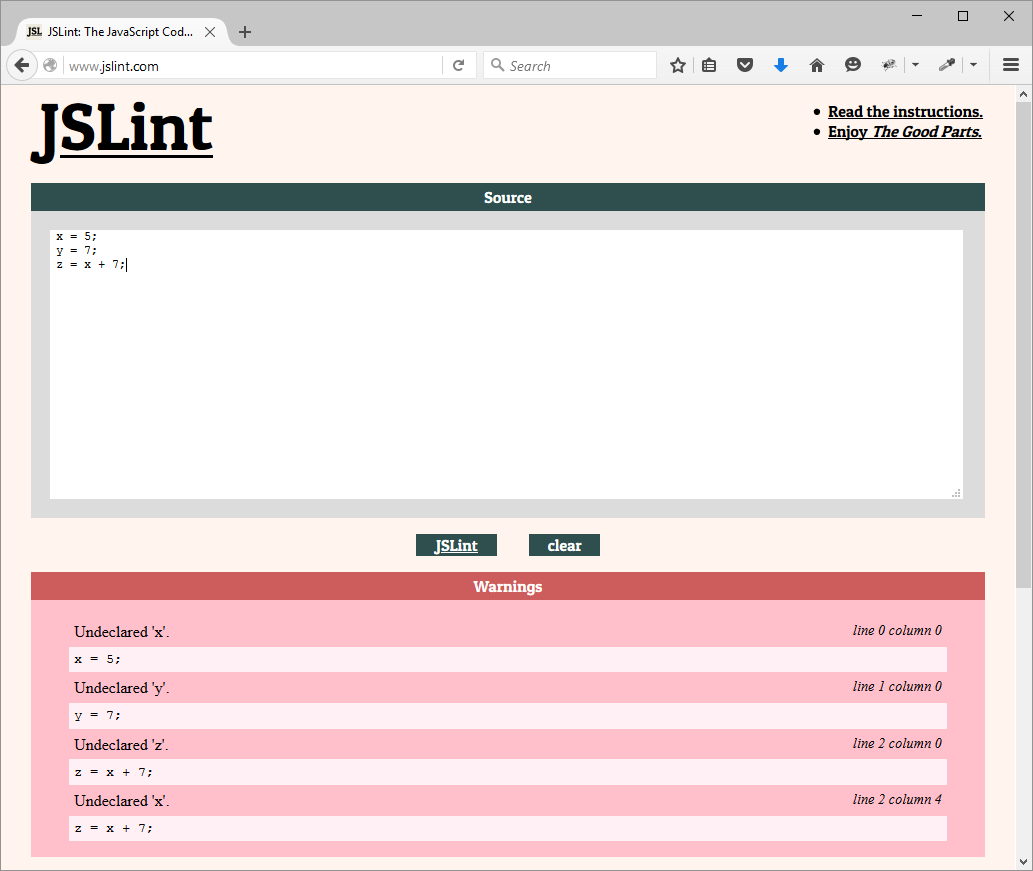


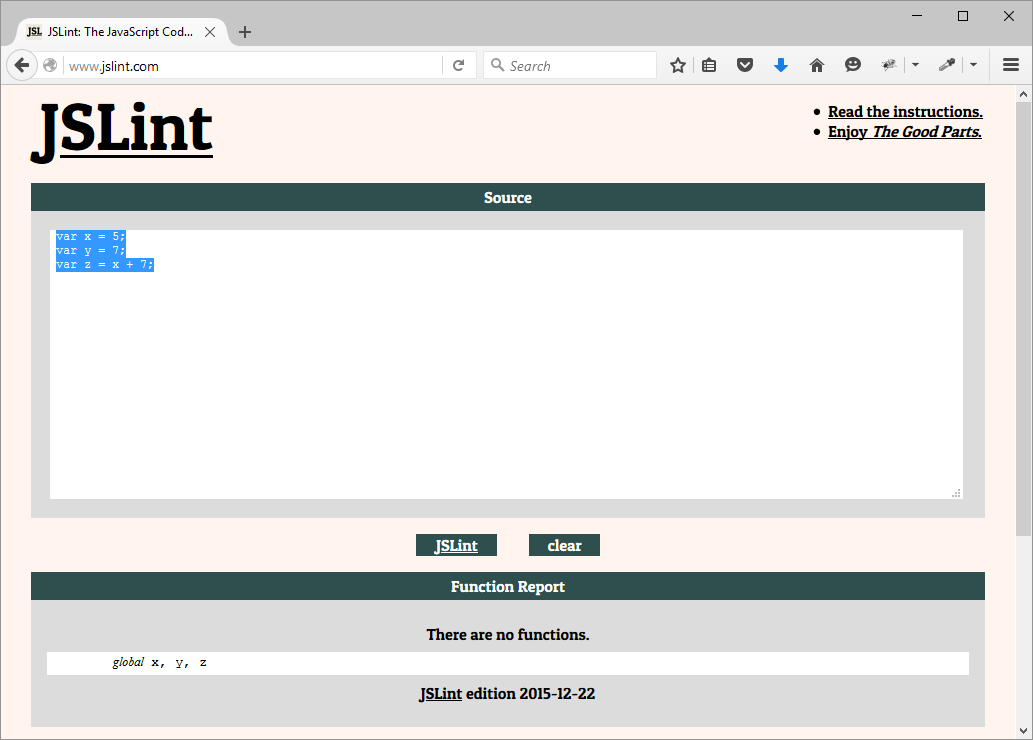
But in fact, lacking the “var” keyword, this has created (probably-unwanted and unneeded) global variables x, y, and z.

# JSLint

A “lint” tool is useful for any computer programming language. It involves holding the programmer to higher coding standards than the bare minimum. Just because a certain language permits certain bad habits and bad coding to occur, doesn’t mean we should code that way.

I’ll try the JSLint.com lint tool:





The morals of the story:

* Use “var” to create variables, almost always
* Use a lint tool such as JSLint to help you code better.

Note: we should be using better variable names than x, y, and z.

# Data Types

There are seven data types in JavaScript:

Boolean,

Null,

Undefined,

Number,

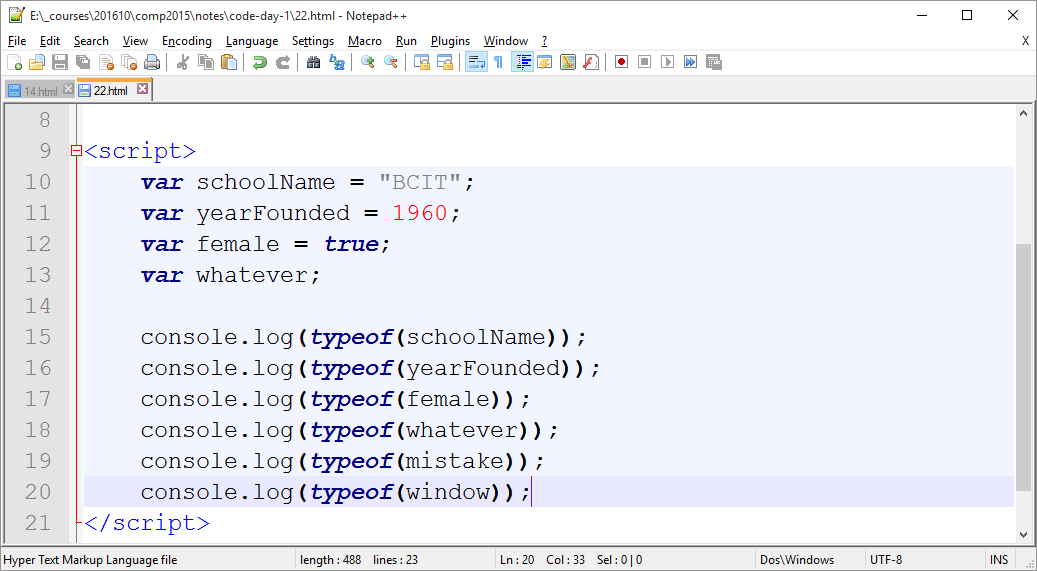
String,

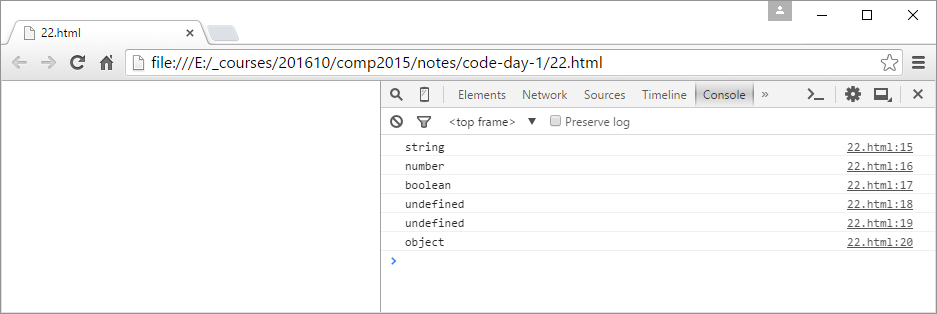
Symbol

(these are all *primitive, immutable* data types), and

**Object (mutable).**

JavaScript is a weakly-type language (also known as dynamically typed): you do not need to specify the data type when you create a variable; and the data type can change when the value changes, too.





# Lab 1

Due next week before class to the course “lab 1” folder on learn.bcit.ca.

Write one .html file and one .js file (named firstname-lastname.html and firstname-lastname.js; use your own first and last names).

The .html file will have three images in it:

a cat

a dog

a bird

The .html file will have three <script> sections in it:

Script 1: in the <head> section which references the external .js file

Script 2: in the <head> section

Script 3: at the end of the <body> section

Script 1 must be deferred. Then it must change the width of the bird image to equal the width of the cat image.

Script 2 must alert the appName of the current browser. Then it must create two variables, x and y, equal to 10 and 20, and add x and y together and alert the sum. Test this script in JSLint first to make sure you do not get any warnings.

Script 3 must change the width of the dog image to equal the width of the cat image.

Important:

The .js file must include a JavaScript comment which says your first and last names at the top of it.