CS 1100: Web Development: Client-Side Coding / Fall 2016
Lab 6: Exploring JavaScript

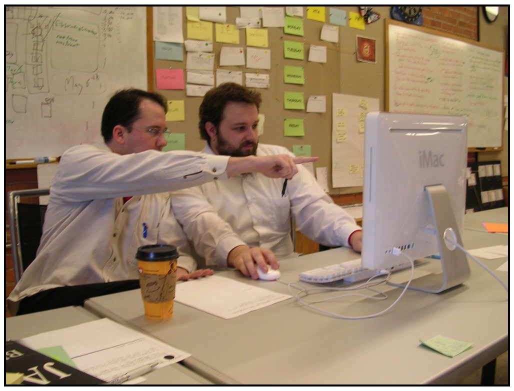
Goals

- Learn to use the JavaScript console interpreter
- Experiment with basic JavaScript statements and expressions
- Create appropriately named variables
- Experiment with values of different data types
- Start using the Document Object Model (DOM) tree to access and modify the contents of a web page

Description

Work in pairs. This lab is a work-in-pairs exercise. Find a partner and work together.

How to work with a partner? Use ONE computer. Both of you will work on each exercise; however, only one can be "the driver" - i.e., the person typing the commands for a given exercise, while the other one is an active participant. You will discuss the exercise and its results together, then switch and let the other person be the driver. You don't have to switch after each exercise - just make sure you spent roughly the same amount of time doing the typing.

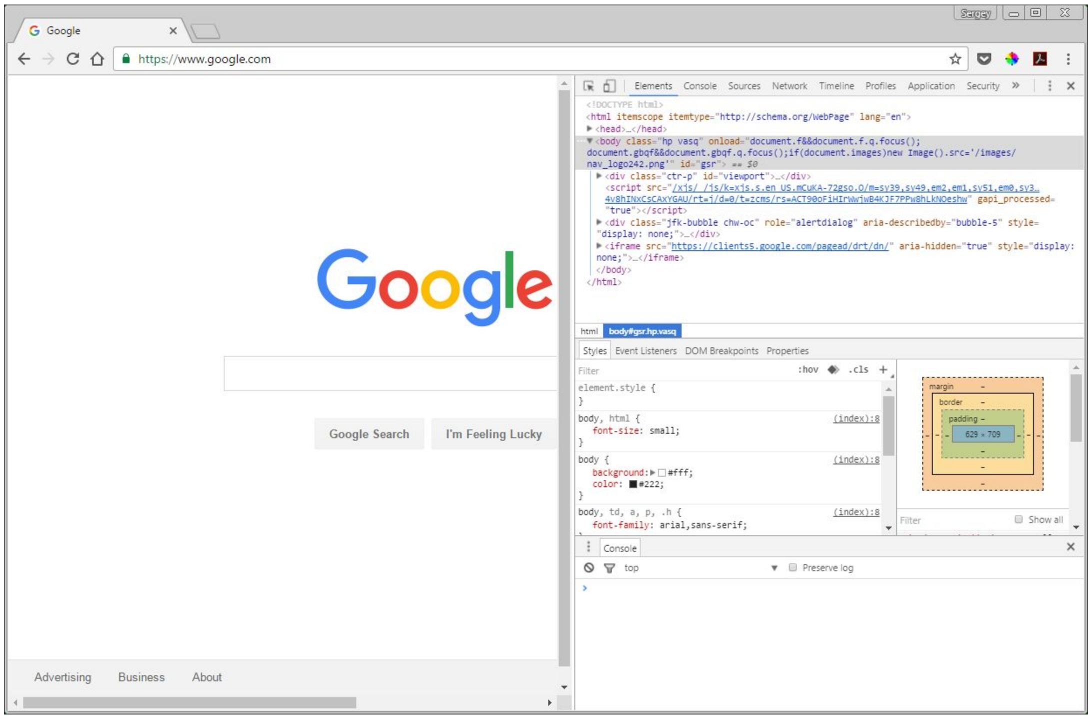


In the real world, this practice is called **pair programming**: "*an agile software development technique in which two programmers work together at one workstation. One, the driver, writes code while the other, the observer or navigator, reviews each line of code as it is typed in. The two programmers switch roles frequently.*" (Wikipedia)

About Google Chrome for today's lab
Today's lab is based heavily on the JavaScript console. We will be using the console available on Google Chrome. You can do the same exercises using any other modern browser (e.g., Firefox, Safari, Internet Explorer/Edge); however, the consoles are slightly different on each of these browsers, so the lab's instructions may not be 100% compatible with the other browsers. The differences are minor and are mostly related to launching and using the console, so if you have to use a different browser and are willing to (quickly) figure out these minor differences, you may do so. Alternatively, pick up a laptop from the cart and use Google Chrome.

Activity 1: Setup your environment

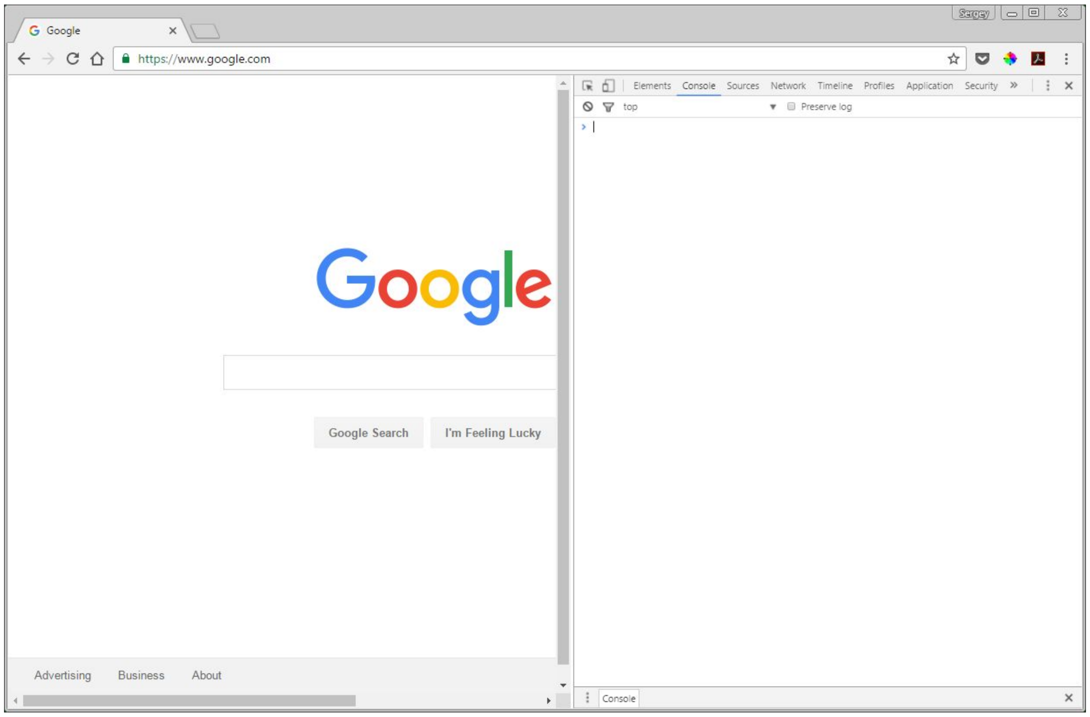
1. Launch Google Chrome. Right-click anywhere on the page and click Inspect. You may also select the Chrome menu at the top-right of your browser window, then select More Tools > Developer Tools. If you were on the google.com homepage, here's wgat you will see (or something very similar):



```
<!DOCTYPE html>
<html itemscope itemtype="http://schema.org/WebPage" lang="en">
 ▶<head>…</head>
 ▼<body class="hp vasq" onload="document.f&&document.f.q.focus();
document.gbqf&&document.gbqf.q.focus();if(document.images)new Image().src='/images/
nav_logo242.png'" id="gsr"> == $0
   ▶<div class="ctr-p" id="viewport">…</div>
    <script src="/xjs/_/js/k=xjs.s.en_US.mCuKA-72gso.O/m=sy39,sy49,em2,em1,sy51,em0,sy3…
    4v8hINxCsCAxYGAU/rt=j/d=0/t=zcms/rs=ACT90oFiHIrWwjwB4KJF7PPw8hLkNOeshw" gapi_processed=
    "true"></script>
   ▶<div class="jfk-bubble chw-oc" role="alertdialog" aria-describedby="bubble-5" style=
   "display: none;">…</div>
   ▶<iframe src="https://clients5.google.com/pagead/drt/dn/" aria-hidden="true" style="display:
   none;">…</iframe>
   </body>
</html>
```

```
element.style {
}
body, html {                                  (index):8
   font-size: small;
}
body {                                        (index):8
   background: ▶ □#fff;
   color: ■#222;
}
body, td, a, p, .h {                          (index):8
   font-family: arial,sans-serif;
```

Notice the horizontal menu at the top of your developer tools. You are currently on the Elements option. Click Console. This will open the console tool, which you can use to execute JavaScript statements interactively. (If you see an error message, click the "clear console" icon in the top left corner of the console - it will remove that message.)



What is interactive? You *interact* with the console by typing in JavaScript statements that are executed as soon as you hit Enter (i.e., the JavaScript interpreter interprets and executes them interactively - one at a time). This is different from writing a program in a text editor, saving it as a .js file, linkin git to a web page, and then see the code executed by opening the web page in a browser of your choice.

Try typing a number in the console and hit Enter - the number will be printed to the console. Now try printing a simple arithmetic expression: 2 + 2. You will see 4 printed: the console has just **interpreted** your code and **evaluated** the expression you typed!

Keep in mind that the console does not require you to use semicolons: each statement is terminated by you hitting the Enter key. This is NOT the case when you will be working with a .js file.

Activity 2. Exploring basic JavaScript

Look at the following statements and try to predict what the result will be. After that, type the statements into the console, each followed by the Enter key. Record the results (i.e, what the console displayed (the console also displays what you type - you may ignore that):

Q1-1. 2

Q1-2. 3

Q1-3. 2 + 3

Q1-4. 2 * 3

Q1-5. 10 / 3

Q1-6. 10 % 3

Q1-7. -10

Now let's try creating some variables. Do the same for these statement (i.e., predict the result, type them in and hit Enter)

Q2-1. var a

Q2-2. a = 2

Q2-3. 2

Q2-4. a

Q2-5. a + 3

Q2-6. a

Q2-7. a = a + 3

Q2-8. a

Q2-9. var b = 4

Q2-10. b

Q2-11. b + a

Q2-12. b

Q2-13. b = b + a

Q2-14. b

Q2-15. a = b

Q2-16. a

Q2-17. z

Q2-18. var z

Q2-19. z = 1

Q2-20. z

There are other operators in JavaScript. Let's look at 2: the increment and decrement operator:

Q3-1. a

Q3-2. a++

Q3-3. a

Q3-4. b

Q3-5. b--

Q3-6. b

As you may have guessed, the increment and decrement operators are shortcuts:

a++ is equivalent to a = a + 1 and a-- is short for a = a -1

Thus, this operator combines an assignment statement with an arithmetic expression where the second operand (i.e., rightmost number) is the integer 1.

There are also operators that combine assignment with basic arithmetic. Try these:

Q4-1. a += 5

Q4-2. a

Q4-3. a -= 1

Q4-4. a

Q4-5. a *= 2

Q4-6. a

Q4-7. a /= 10

Q4-8. a

All these operators are useful: they keep you code clean and concise.

Finally, let's try something different.

Q5-1. var text = "the answer is "

Q5-2. text

Q5-3. var newText = text + 42

Q5-4. newText

Q5-5. var quantity = 5

Q5-6. var result = quantity + 6

Q5-7. result

Q5-8. result = result + "7"

Q5-9. result

What just happened?
When we add numbers, the result is a number.
When we add strings, the result is a string. We call this string concatenation.
When we add numbers and strings, the numbers are converted to strings - so in the last case, you were not adding the number 7 - you were adding the character 7 (notice the quotation marks) - and, therefore, the other operand - the number 11 (current value of result) was converted to a string (characters 1 and 1).

Activity 3. Exploring the Document Object Model (DOM) tree

Download the lab6.zip file and extract its contents. You will see an HTML file lab6.html and a folder resources. The folder contains CSS files and a few images used on the web page. We will not use these files today, but we need them for the web page to display correctly. We will only work with lab6.html.

Open lab6.html in Google Chrome. This is slightly edited version of the notes from session 2. We will use this file to experiment with the DOM.

Use the built-in document object and its method querySelector() to access HTML elements on the page. You may then use the innerHTML and innerText properties to access and modify the text or HTML of the element you access.

Here's an example:

The following code retrieves the first occurrence of an H1 element and assigns it to variable heading1

```
var heading1 = document.querySelector("h1")
```

The following code accesses the text content of this element:

```
heading1.innerText
```

which you can assign to a variable:

```
var headingText = heading 1.innerText
```

You can also change the element's text:

```
heading1.innerText = "new heading!"
```

You can also assign the text content of one element to another element. Try to accomplish the following:

Q6. Switch the text values of H1 and H3 on this page. Hint: ***you will have to*** use variables.

Q7. Try to modify, ideally switch the innerText properties of 5 more elements and record your results.

Q8. Try the same using the innerHTML property. What's the difference?

Q9. Use the built-in prompt() function to get a value from the user AND assign it to a variable. Let's ask the user for a valid CSS selector that applies to this webpage:

```
var selector = prompt('provide a valid css selector for this page')
```

Notice that the text in the quotation marks is a string which is displayed to the user as a prompt - so we can use any text we like.

Now use the same approach to get new text from the user:

```
var newContent ' prompt('what should be the new content?')
```

Now check what your variables contain (type and hit Enter after each one):

```
selector
```

```
newContent
```

Now find the element using the user-supplied selector:

```
document.querySelector(selector)
```

Notice that we DID NOT use quotation marks around the variable name: it is NOT a string!

We can improve our last statement by assigning the retrieved element to a variable:

```
var myElement = document.querySelector(selector)
```

And now, of course, we can change its text content!

```
myElement.innerText = newContent
```

Let's put it all together:

```
var selector = prompt('provide a valid css selector for this page')
var newContent ' prompt('what should be the new content?')
var myElement = document.querySelector(selector)
myElement.innerText = newContent
```

You've just given the user the ability to modify ANY part of your web page. Congratulations!

Submitting your work

You need to submit only ONE response sheet. Make sure to indicate your names on the sheet.