

JavaScript I:

JavaScript Hard

Lesson Plan

In this class we'll cover variables, expressions, and functions. Students need to have intermediate computer skills and a basic understanding of HTML and CSS to attend.

Lesson Objectives

At the end of the class, the student will:

* Understand what a variable is and be able to both declare and initialize a variable
* Understand the basic data types in JavaScript
* Be able to use expressions to do basic math and concatenate strings
* Be able to write and call a function, with and without arguments
* Understand the concept of global and local scope

Lesson Prep Work

(30 min, at a minimum, prior to student arrival)

* get in early to test for technology failure, because it will happen :-)
* create a JSbin account to keep track of exercises
* print handouts

Lesson Prerequisites

● It is recommended that students have some knowledge of HTML and CSS.

Lesson Outline

The lesson is completed in one 120 minute class session.

*(x)* Introduction

* Introduce instructor, students.
  + Ask students at introduction: Have you used JavaScript before? Why do you want to learn JS?
* Let students know it’s okay to take phone calls, but ask them to put their phone on vibrate and answer calls outside the classroom.
* Inform students that they can sit back and watch if the class is too advanced.
* Inform students they can go to the bathroom, they don’t need permission.
* Show order in which class will happen. Explain scope of class.

*(10)* A little background

* JavaScript is a programming language
  + *Explanation*
    - (slide 2) JavaScript is most commonly used inside websites to manipulate HTML, pull in data, and do calculations.
      * In short, JavaScript makes websites do cool things: image lightboxes, web apps like Grooveshark, animation, games
      * Some neat examples:
        + <http://sidigital.co/>
        + <http://www.amplifon.co.uk/interactive-ear/index.html>
        + Pandora.com
        + Denverlibrary.org
    - JavaScript IS NOT JAVA. The two have nothing to do with each other – the fact that they have similar names is due to a marketing campaign run by Netscape (the people who created JS) to compete with Java, which was the hot language at the time.
    - (slide 3) A programming language is just a set of instructions to tell a computer to do something.
    - Each instruction is called a “statement”. Let’s write our first statement.
  + *Activity: Have class write a hello world statement*
    - Step 1 – (slide 4) Have class open jsbin.com
      * Clicking on the bar at the top of the window, close everything but the JS and Console windows
      * *Explanation:*
        + on the left side is where we write our instructions, and on the right hand side is our console, which shows us the results.
    - Step 2 – (slide 5) On the left side, type in console.log(“Hello world!”);
    - Step 3 – Click the “run” button on the top left of the left hand side.
    - The class should all see “hello world” show up in the console. If they don’t, have them check their code against yours and try it again.
* (10) JavaScript syntax
  + *Explanation*
    - (slide 6) How we use symbols and words in a language is called the “syntax” of the language.
    - Look at our first statement – what can we tell about JS’s syntax?
      * JS is case-sensitive.
      * Statements end with a semicolon.
      * That “hello world” is what is called a string – a type of data that is just any amount of text, and can contain letters and numbers. Strings have to be placed inside of quotation marks. They can be single or double, but you should be consistent.
    - Here’s some other things that will be useful:
      * Strings can’t have multiple lines – that is, you can’t hit enter in the middle of a text string to start a new line. JS thinks that you just forgot to finish the string, and you will get an error
        + *Activity: Show class what this looks like on screen* (i.e., add hit enter in the middle of “hello world” and try to run the program)

*Teachers Tip:* point out that the color coding in JSbin will also give students a hint – if everything turns red, it’s still a string – or if it’s not red, JS isn’t reading it as a screen.

* + - * (slide 7) You can also leave comments
        + For a single line, place // at the beginning
        + For multiple lines, use /\* at beginning and \*/ at end.
* (40) Variables
  + *Explanation*
    - (slide 8) We’ve made the console say hello. Let’s learn about variables.
    - Variables are used to store values. Say we want to remember the name a user typed in, or a place they clicked, or the result of a calculation. We store these values in a variable.
    - Variables are “declared” by using the keyword var. To declare means to say to the computer – “there is a variable named ‘X’”
    - Variables are “initialized” when they’re given a value. We give a variable a value by writing its name, an equal sign, and then the value.
      * *Metaphor:* declaring is a bucket. Initializing is filling that bucket with something.
    - (slide 9) Rules for naming variables:
      * They must begin with letters, underscore, or a dollar sign. They can use numbers after that.
      * Variable names are case-sensitive.
      * It’s not required, but considered standard to use camel case (instead of underscore) when naming multiple-word variables (aReallyLongName as opposed to a\_really\_long\_name).
      * Variable names also can’t use any reserved words – words that JavaScript has already assigned special behavior.
        + *Activity:* Show list here: <https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Reserved_Words>
  + *Activity: create a variable*
    - Step 1 – (slide 10) Declare the variable firstOne: in JSbin, write var firstOne;
    - Step 2 – Set a value for firstOne: on the next line, write firstOne = 1234;
    - Step 3 – To see the value of our new variable in the console, let’s use console.log: on the next line, write console.log(firstOne);
    - Step 4 – Run the program.
    - Step 5 – (slide 11) Show that we can declare and initialize at the same time. Add var secondOne = 5678;
    - Step 6 – Ask class to add a line which writes the value of secondOne to the console (console.log(secondOne);)
  + *Explanation*
    - (slide 12) Once we’ve declared a variable, we can change its value at any time. We skip the var keyword and simply assign it a new value!
    - Point out that we didn’t put quotation marks around the values for our variables in the exercise. This is because they are different data types,
    - (slide 13) There are five “primitive” data types in JS:
      * Strings: a bunch of text.
      * Numbers: can be whole or floating point
      * (slide 14) Boolean: true or false
      * Undefined: when a variable has been declared but hasn’t been assigned a value
      * Null: empty, but not unassigned
      * Everything else is an object
    - (slide 15) JS is a loosely typed language – you don’t have to declare data types when you declare a variable, and a variable can be changed to any data type by giving it a value of that data type.
  + *Activity: find the data type of a variable, change data types*
    - Step 1 – (slide 16) to existing code in browser, add console.log(typeof firstOne); and hit run.
    - Step 2 – reassign firstOne to “now I’m some text”
      * *Teacher’s Tip:* Point out how JS errors out if you stick with single quotation marks – we have to use the double quotation marks if our string has a single quotation mark inside.
    - Step 3 – copy and paste console.log below reassignment and run. Students should see data type change.
* (25) Expressions
  + *Explanation*
    - (slide 17) Variables can also contain an expression – the result of a piece of code that evaluates to a value.
      * (slide 18) Can use basic math operators + (addition), - (subtraction), \* (multiplication), / (division).
        + %: the modulus is for remainders – it returns any remainder from dividing two numbers. So 4%2===0, but 5%2===1.
        + There’s also shorthand for adding or subtracting one from a number. The increment (++) adds 1 and the decrement (--) subtracts 1.
      * (slide 19) When used with text strings, the + will concatenate – combine the text strings into one large string.
        + That is, 8+3=11 but 8+“is my favorite number=“8 is my favorite number”
  + *Activity: use some expressions, see loose typing in expressions*
    - Step 1 – clear out bin
    - Step 2 – add the following expressions:
      * var x = 42 + 37;
      * var y = 356 - 204;
      * var total = x + y;
    - Step 3 – ask class to find out what these evaluate to. (They can use console.log)
    - Step 4 – (slide 20) clear out bin. Have the class add the following variables. Note that they are two different data types:
      * var x = "8";
      * var y = 3;
    - Step 5 – ask class to find answers and data type for following expressions:
      * X+y (83, string)
      * X-y (5, number)
      * X\*y (24, number)
      * x/y (2.66, number)
      * Bring up slide 21 after several minutes to help people out.
      * *Teacher tip:* (slide 22) after they’ve puzzled by themselves for a bit, show class how get both answers at once. One way is console.log(‘The value of x+y is ‘ + x + y + ‘ and the data type is ‘ + typeof(x+y))
        + Also – remind students for output to be spaced properly, they must include spaces in the strings.
  + *Activity: solo exercises – have classes work out answers by themselves, then discuss solutions as a group.*
    - Exercise 1: (slide 24) age calculator
      * Store the current year in a variable.
      * Store their birth year in a variable.
      * Calculate their 2 possible ages based on the stored values.
      * Output them to the console like so: "They are either NN or NN"
      * *One solution:*var year = 1976;  
        var now = 2014;  
        var age = now - year;  
        console.log('They are either ' + age + ' or ' + (age - 1));
    - Exercise 2: (slide 25) geometry
      * find the circumference and area of a circle.
      * Store a radius into a variable.
      * Calculate the circumference based on the radius (pi\*diameter), and output "The circumference is NN".
      * Calculate the area based on the radius (pi\*r\*r), and output "The area is NN".
      * *One solution:*var radius = 3;  
        var circumference = Math.PI \* 2\*radius;  
        console.log("The circumference is " + circumference);  
        var area = Math.PI \* radius\*radius;  
        console.log("The area is " + area);
* **(20) Functions**
  + *Explanation*
    - (slide 26) Functions are reusable collections of statements – if we have to do the same thing again and again in a program, we can write a function and then call it.
    - Functions are declared, just like variables, but with the name “function”. Give it a name, two parentheses (more on that in a second) and then we have the curly brackets around the statements that make up our function.
      * We don’t have a semicolon at the end of the function – you don’t put one on a compound statement.
    - Functions aren’t run right away – they’re stored in memory until you call them. We call a function by writing its name.
    - (slide 27) Functions can also have arguments – values that the function needs to return a result. These go inside the parentheses. You can have one, or many separated by commas (slide 28).
    - Coding conventions – use new lines between statements and indent to show blocks.
  + *Activity: write a function that writes someone’s name to the console*
    - Walk them through the following:  
      function sayMyName(name) {  
       console.log(‘Hi, ’ + name);  
      }
    - Have the class call the function at least three times with different arguments. i.e.:  
      sayMyName(‘Ted’);  
      sayMyName(‘Ned);  
      sayMyName(‘Fred’);
  + *Activity: write a function that calculates a sum*
    - Walk through the following:  
      function sum(num1, num2) {  
       var result = num1 + num2;  
       console.log(result);  
      }
    - Have the class call the function at least three times with different arguments.
  + *Explanation*
    - (slide 29) You can also pass variables to functions as arguments.
      * *Activity:* As demonstration, have class make a variable, assign it a numeric value, and use it as an argument in the sum function:  
        var smallNum = 3;  
        var bigNum = 29;  
        sum(smallNum, bigNum);
    - *Discussion:* (slide 30) The “return” keyword sends the result of a function back to whoever called it. Why would we need this?
      * Try assigning the sum function as the value of a new variable, like this: var bigSum = sum(42,123);
      * Console.log bigSum. You’ll receive the answer “undefined” – our sum function sends an answer to the console, but not to who called the function – in this case, bigSum - so bigSum is “undefined”.
      * (slide 31) Add a return to the sum function and try again.
    - Remind class that return exits the function – anything after it won’t be executed – so best to put return at the end.
      * Can demonstrate by adding a console.log after the return and seeing if it runs.
* **(15) Scope**
  + *Explanation*
    - (slide 32) JS variables have function scope – they are only visible in the function in which they were declared.
    - So a variable like result in our sum function has “local scope” – it’s not accessible outside of the function.
    - Variables declared outside of a function are visible to everyone – they have “global scope”
  + *Activity: demonstrate local and global scope with the sum function*
    - (slide 33) Have class console.log the result variable – they’ll get the message “ReferenceError: localResult is not defined”
      * Point out how that call is outside the function, and thus can’t access the local variable.
    - (slide 34) Now move the variable declaration outside of the function, like this:

var result;

function sum(num1, num2){

result = num1 + num2;

console.log("The local result is " + result);

}

sum(4,2);

console.log(“The global result is “ + result);

* + *Activity: functions wrap-up*
    - Have class repeat age calculator exercise, but now write a function to do the work for them. Answer should be something like this:
      * function calculateAge(birthYear, currentYear) {  
         var age = currentYear - birthYear;  
         console.log('You are either ' + age + ' or ' + (age - 1));  
        }

**(x)** **Conclusion**

* Go over handout, review material, and emphasize contact info & further resources on handout.
* Any questions? Final comments?
* Remind patrons to practice; assign take-home-practice - remind them they can ask for help
* Remind to take survey.

Appendix

Key Decisions

1. I’ve decided to use jsbin in class, largely because you can set the text size without having to zoom in the whole screen. There are many other tools you can use, including Codepen and JSbin.

Links

Teaching Materials: http://www.teaching-materials.org/javascript/

Oz Girl Develop It: http://cathylill.net/courses/js101/

Girl Develop It: http://www.girldevelopit.com/materials