| Sub-Section # | Assignment Title | The task | Assignment outline |
| --- | --- | --- | --- |
| 4.11 | "Setting up your GitHub" | * + Go through Github's guided signup process   + Customize your Github profile | The student creates a GitHub profile, a readme.md with some personal & fun factoids, and uploads a picture (if they've already created a LinkedIn page, etc. previously, they should use the same picture for personal-brand-consistency). |
| 4.12 | "Make a fork from our Repository" | * + Go to Bitdegree's github repo and fork it   + Clone the repo to make it local   + Run the repo's html file to get an unexpected output   + Open the javascipt file to fix the problem   + Re-run the html/refresh the webpage to ensure that the problem is fixed   + Git add, commit and push to the student's forked repo | The student has been hired at their first job and needs to download the code base in order to start fixing problems … in the given example, we have   * A commented line that needs to get uncommented, so that * We can see the “hidden” message in the web browser terminal |
| 6.5 | "Link script.js to index.html" | * + Create an index.html file   + Insert script tags and run a console.log command   + Create a script.js file, cut-paste your code from index.html to script.js, and update script tag with "src" attribute   + Test that your code still runs | * + The student needs to prep for creating their personal/portfolio website, beginning with an index.html page   + They are also going to need to create a script file in order to do anything visually fancy (or alter things included in a template website) |
| 6.6 | "Console.log your name" | * + Console.log your name in the browser's console   + Console.log your name in terminal | * + Students need to console.log their name, and   + (sign-posted) The result of a simple math equation, and   + (sign-posted) Checks to see what lines are running in Javascript |
| 6.7 | "Use single-line and multi-line commenting" | * + Add three additional console.log commands to script.js file (your birthday, your favorite pet's name, & the city you were born in etc)   + Comment-out the second one using the single-line comment ("//") and run your code to test that the second one doesn't fire   + Un-comment-out the second one and instead comment-out your console.logs #2-4 using the multi-line ("/\* \*/") | * + Make a note to yourself (that is one sentence long) about changing one of your lines of code using in-line commenting   + Make a note to yourself (that is a paragraph) listing the first three projects you'd like to code in JS, and describing them), and then comment this out using the multi-line comment symbols |
| 7.12 | "Write primitive Data types" | * Write three strings:   + A basic string   + A string that contains a literary quotation from an author/quote from a public speaker (perhaps using interpolation)   + A single string that spans multiple lines of code | * An autoparts supply company needs you to update their website’s ‘Welcome’ page to improve business, as it currently just has the company logo (and no text). * Console.log three strings we might add to the page:   + A basic greeting, like     - console.log('Hello valued customer!');   + A more personal version of that greeting, like     - console.log('Hi Neighbor! You know what we always say: "We have better prices than our competition - \*every\* time!"');   + A greeting that also includes a “call to action” to browse the site … maybe on multiple lines, so that it is more readable:     - console.log(`Hi Neighbor!   You know what we always say: "We have better prices than our competition - \*every\* time!"  And make sure to check our “latest discounts” page!`); |
|  |  | * Use three methods | * Try punching up our first line (“Hi Neighbor! …) by making the entire line capitalized * Check to see how many elements/characters are in the second line (what’s the “character count”) * Split the second line into two, at the “:” character so that it reads   + You know what we always say:   + “We have better prices than our competition - \*every\* time!" |
|  |  | * Perform a basic math calculation * Check to see that your result is a number and not a string , using typeof method | * The auto-parts supplier would like for customers to see a welcome page that celebrates how long they’ve had an account for, or note the day they first signed-up.   One way to solve this is by subtracting the number of days since they first created their account from the current date, so (assuming we’re in the month of September), subtract 27 from 31 (using console.log) in order to find out that the account was created on the 4th * Check that what’s being returned is a number and not a string or some other data type! |
|  |  | * Perform a boolean check | * Using console.log, check to see whether 27 is greater than 31 or whether 4 is less than 1 - checks you might use later to make sure your code is calculating the correct month, and then... * Perform another check to make sure the element you’re returning (4) is not a string, using === |
|  |  | * Declare a variable to store this information, and then check that it is empty - that it is “undefined” | * Create a variable like customerSignUpDate or customerVintage to store this information (with appropriately-declared editability) and use console.log to check that your variable has 1. been created, but 2. Is currently empty (returns “undefined”) * Put your result (4) in your variable and then re-check the variable’s contents to make sure your number was stored successfully (and doesn’t return Null) |
|  |  | * Test Javascript’s coercion ability | * Using console.log, add your variable to a string   + console.log(customerSignUpDate + “is the first day you joined!”) * Check the datatype of this product using typeof * Now, mathematically add a string to your number   + console.log(customerSignUpDate + “27”) * Check the datatype of this result, too |
| 8.7 | "Variables" | * Declare the variable * Initialize the variable * Re-define the variable | * After being contracted by Amazon to update their internal database code, you find that you will need to create editable variables in both older and newer formats, as well as unchangeable/permanent variables. |
| 8.8 | "Template literals" | * Use a template literal (and one or more substitutions) to perform interpolation of   + A variable into a string   + A mathematical equation into a string   + An evaluation into a string * Add “use strict” to your file and re-run your code to make sure it still works / to see if you are developing any natural, but bad habits | * Using either autoparts supplier file, or your Amazon Database file, perform several interpolations:   + Add a smaller string into a larger string at a specific position using a template literal - say, by     - storing the customer’s name in the variable **customerName**, and then     - console.log-ing(`Hello **${customerName}**! You know what we always say …`)   + Performing math:     - 'Hi ${customerName}!   **It’s been ${31-4} days since you joined our family - and**  you know what we always say: "We have better prices than our competition - \*every\* time!"  Make sure to check our “latest discounts” page!`);   * + `Performing an evaluation on a variable we haven’t initialized yet:     - 'Hi ${customerName}!   It’s been ${31-4} days since you joined our family - and you know what we always say: "We have better prices than our competition - \*every\* time!"  Make sure to check our “latest discounts” page!  **${console.log(dateJoinedVariable)}**`); |
| 9.7 | "Equals" | * Test equivalence (type) * Test strict equivalence (value *and* type) * Test *not* equivalent and *strict* not equivalent | * Create four variables, two of which are numbers, and two of which are string-versions of those same numbers   + Test whether 1 == “1”, whether 1 === “1”, whether 1 == “2”, and whether 2 != “1” * Create two additional variables that store words   + Ex: Test to see whether “about” is the same as “About”   + (if we expand scope to cover greater than etc) Test whether “about” is greater or less-than “boat” |
| 10.5 | "If … else if … if statement" | * Test an “if” statement * Test an “else” statement * Test an “else if” statement * Test a ternary statement | * Pizza order ... Alocal Pizza restaurant has started a rewards program, and they need you to begin building out this part of their website. Their customers will be able to collect points after every order and use them to receive tasty rewards. So ...   + create a variable for the cash-points balance a user has. Then ...   + Write a conditional statement using “if” and “else(s)” where …     - If you have less than 100 points, console.log that “You don’t have enough for a complementary pie, yet - please come back later!”     - If you have between 100 and 200 points: “Please choose any small pizza on us!”     - If you have between 200 and 300 points: “Please have a medium pizza of your choice for free!”     - If you have more than 300 points: “Any large pizza is on the house - thank you for being such a loyal customer!”   + The owner is considering a special, secret bonus, so also write a ternary statement that checks to see whether the balance is over 1,000 points -     - if it is, console.log “Congratulations! We would like to invite you to a special dining experience - please call with your account info handy     - If it is not, say “By the way - we love having you as a customer. Please keep coming back for something special …” |
| 10.6 | "Switch" | * Write a switch that includes at least two “cases” and a “default” case (plus “breaks”) | * The local Pizza place would like their website to greet people visiting their website with unique messages for each day of the week:   + Create a weekDay variable, store a day of the week in it, and   + write a “switch” that returns a different string statement for each day of the week (with breaks). Make sure that you include breaks for each day’s case, *and*   + define a “default” case (which will cover all other inputs), alerting whomever maintains the site that something is wrong (like, “Something is funny about today” or “What day of the week is it?” etc) |
| 11.4 | "For loops" | * Write a “for” loop | * Create a ‘for’ loop that prints to console “99 bottles” (and/or ...) * A job-interview/hackerrank nested-for-loop problem |
| 11.6 | "For … of" | * Write a “For … of” loop | * Anauthor has just finished their second manuscript, and has asked you to help them write a program that can automate some of the editing. In preparation, create an algorithm that can re-format a poem into a multi-line string:   + Declare a variable, initialized with a poem like “This Little Piggy Went to Market” - something with verses separated by commas   + Declare a second variable as a string (so we can store the result of our work   + Create a For...of loop that  1. Declares an internal variable 2. Targets the original poem, and 3. Uses the .split method to separate lines by comma (Hint: .split(“,”) 4. Console.log logs the internal variable each time (to make sure the function is working properly) 5. Stores the contents of the internal variable in our second, external “result” variable *with* a line break (Hint: + ‘\n’) 6. And then console.log logs our “result” variable once the For..of loop has completed |
| 11.5 | “While loops” | * Write a “while” loop | * An early education company wants to add a new math module to their tablet suite, and they’ve asked you to create a game that challenges children to do basic addition.   + Declare a variable that stores five tries   + Create a ‘while’ loop that, as long as the number of tries is still greater than 0, will:     - Create an object/internal variable to store the result of a prompt/user input,     - Prompts the child to solve a simple addition problem (5 + 2 etc), and tells the child how many tries they have remaining,     - Use a conditional/’if’ to check whether said object equals the correct answer (remember to only compare value, and not type) -       * If the input is correct, a “congratulations!” message is displayed       * If it is not, the “chances” variable is decreased/deprecated by 1   + Create a “Let’s start over and try again!!” message to display when the while loop finishes |
| 12.5 | "Manipulating array items" | * Use .push, .pop, .unshift, and .shift methods on an array | * You’ve volunteered to organize a surprise birthday dinner for your best friend and need to get a headcount on who will be attending.   + Create an array of names you ***think*** will be going     - Ex: let friends = [“Sarah”, “Lukas”, “Claude”, “Annette”, “Matthias”]   + Next, add “Darius” to the list, using the .push method   + You discover your friend’s sister will be in town, and ***must*** be included - use the .unshift method to make sure she’s first on the list   + You just got a text on your phone from Darius “I’m not feeling well.” Remove him from the list using the .pop method |
| 12.6 | “Iterating arrays using 'for..of' loop” | * Write a ‘for..of’ loop | * An advertising company wants to target the adult demographic on their mailing list, and they need you to identify only those people over 20 years old. So, * Create a ‘for...of’ loop that   + Reads the following declared list:     - let subscriberAge = [16, 22, 38, 45, 33, 17, 27, 55, 17]   + Console.log’s what’s been stored in the loop’s object/variable,   + Uses an ‘if’ statement to check whether this element is over 20 - and if it is, store it in a new variable that can be accessed after the ‘for’ loop is finished |
| 12.7 | “Iterating arrays using 'for' loop” | * Write a ‘for’ loop? |  |
| 12.8 | “Iterating arrays using 'forEach' loop” | * Write a ‘forEach’ loop? |  |
| 13.3 | "Using Array methods" | * Use the following methods:   + .indexOf   + .lastIndexOf   + .splice   + .slice   + .concat   + .join   + .split (a string method) | * The program you’re writing for your author-friend needs some additional features. Using the provided text, we need to:   + We need to find where a specific line starts (using the .indexOf method)   + We need to find something later in the manuscript (using the .lastIndexOf method)   + We need to grab a quote and store it in a new variable (using the .slice) method   + We need to delete a phrase (using the .splice) method   + We need to combine several pieces of information into a sentence/string (using the .join method)   + We need to add that sentence at the end of the manuscript (using the .concat method) |
| 13.4 | "Using different array methods" |  |  |
| 14.2 | "Sprint 8 Project" | Setup:   * Create a folder to store their project in & create a blank/starting js document in this folder * Create a git repo, add their local project to this repo, comment, & push the folder’s contents to github * Check that the push succeeded   Phase I:   * Use strings, numbers, booleans, variables and arrays * Use template literals * Use equivalency and mathematically-evaluative checks * Use conditionals * Use for loops * Use while loop * Use string & array methods * Cement Best Practices, including comments, frequent console.log-ing & git committing   Phase II   * Final Commented Code: add section titles and clarifying notes where needed; remove scratch notes etc in preparation for “production” push   Final   * Push “production” code & check that it pushed successfully | * “Mad Libs” Project   + Declare and initialize a variable that stores a mad libs paragraph   + Create a loop that runs through the string and, at every “\_” space (or equivalent “blank” character) solicits an input from the user |
| 14.3 | "Project Explanation" |  |  |