|  |  |
| --- | --- |
| Sprint 9, Assignment x Please also update the doc name with correct numbers. | |
| Assignment type: JS Interactive | |
| Assignment name  Can remain the same as the assignment topic name, or…  can be created depending on the task in the active form, e.g. “Create your own X” | “Exit Project” |
| BDG Description\*  What is the task and why is it important?  In this part, it’s encouraged to think about storytelling and future job-specific context e.g. “You’ve been asked to help out X with Y. They want Z on their website, yet aren’t too sure on how to achieve it”.  Drawing on practical examples and adding context can increase a student's motivation and increase long term learning according to Instructional Design principles, because this helps to relate some familiar or existing knowledge to new bits of information.  This will appear in the course as text before a button, leading to the interactive platform assignment. | Using what you have learned so far, it is time to create a basic encryption algorithm! Try to create this algorithm using 1) a hash-object, and 2) a loop. If you are successful, your code should convert a normal message into a secret message!  Bonus points if you can make your code convert not just letters and numbers, but also full sentences into secret messages! Even more points if it can turn encoded message \*back\* into a regular ones, and gold star, A+ points if you can turn your loop into a function! … be the cyber ‘White Hat’ you were always meant to be! |
| The Assignment  A short specific description of the assignment and tasks using bullet points that the student will need to do. |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Steps  Step-by-step instructions on what the student should do. | | |  |
| Step # | Step  Write each small step of the task | At least 1x hint(s)  Write some text (not necessarily, but can also be a part of code if relevant) which would hint the student to figure out the correct step forward. | The correct output should be…  (if relevant to the task) |
| 1 | Log to terminal a test sentance to make sure everything is set up for develoment |  | console.log(`Let's create a basic encryptor!`); |
| 2 | Create an object with keys of all 26 letters in the english alphabet, and values of all 26 letters in reverse (so, your first property will be "A" : "Z", your second will be "B" : "Y", and your last will be "Z" : "A") |  | const converter = {          "A" : "Z",          "B" : "Y",          "C" : "X",          "D" : "W",          "E" : "V",          "F" : "U",          "G" : "T",          "H" : "S",          "I" : "R",          "J" : "Q",          "K" : "P",          "L" : "O",          "M" : "N",          "N" : "M",          "O" : "L",          "P" : "K",          "Q" : "J",          "R" : "I",          "S" : "H",          "T" : "G",          "U" : "F",          "V" : "E",          "W" : "D",          "X" : "C",          "Y" : "B",          "Z" : "A",        } |
| 3 | Create a user prompt that will store a letter or word in a variable (Note: this variable is going to get updated, so declare accordingly) |  | let userInput = prompt(`Please enter a phrase or word to convert it into secret code! `); |
| 4 | Log this variable to terminal and test that you're successfully storing the user's input |  | console.log(typeof userInput); |
| 5 | Change the stored user input to a) all capital letters - so it can be run through your object later, and b) split into individual letters - again, so it can be fed one letter at a time into your object. |  | userInput = userInput.toUpperCase();      userInput = userInput.split(``); |
| 6 | Try updating your terminal message to something more readable - say, `This was your input: ${userInput}`. Next, display only the first element of your array (which was created using the .split method) |  | console.log(`This was your input: ${userInput}`)      console.log(`This was your input: ${userInput[0]}`) |
| 7 | In order to keep your code clean, combine your two methods into one line of code (and comment-out your old ones) |  | userInput = userInput.toUpperCase().split(``);      console.log(`This was your input: ${userInput[0]}`) |
| 8 | Create a variable to store our output array - named something like "encryptedArray" (Note: this variable is going to get modified when we re-join the individual elements, so declare accordingly) |  | let encryptedArray = []; |
| 9 | Create a loop that a) will go through each element of your userInput array, b) use that element as a key, and find the associated value in our "converter" object, and c) store that value in our new output variable | Note: your loop should go through your array ... and not the object  Note: a 'for' loop is a great option here, but you can also use other types of loops - after you've succesfully completed this project, try writing some other loops you've learned so-far! | for(let i = 0; i < userInput.length; i++){          encryptedArray.push(converter[userInput[i]]); *//  Here we are calling the letter from userInput[i] in our converter - `Converter - what is associated with "A" or "B" or "C"?`*      } |
| 10 | Log your new, encrypted array to see whether your loop is working properly |  | console.log(encryptedArray); |
| 11 | Re-join your elements into one word and log it to check your end-result |  | encryptedArray = encryptedArray.join(` `);      console.log(encryptedArray); |
| 12 | Bonus: make your algorithm work for sentences, in addition to letters and words | Hint: your object, currently, doesn’t know what to do with spaces |  |
|  |  |  |  |
| ... |  |  |  |