|  |  |
| --- | --- |
| Sprint 8, Assignment 1.9 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” | “Create an Object with Constructor Function” |
| 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. | Santa wants to know if you’re ready to move forward on the project. Create a few files for potential stops in late December, and add an address property. Then, compile a list based on who has been nice (and will be getting a visit). Note: at least one child must be designated as “bad” (using a boolean value) and also have provided 0 cookies last year - though representatives apparently can neither confirm nor deny any connection. |
| The Assignment  A short specific description of the assignment and tasks using bullet points that the student will need to do. | * Create an object with properties * Add an address property, and store its value as an array of one or more strings * Create four instances with values - at least one of whom has been bad * Use an ‘if’ statement on naughty/nice |

|  |  |  |  |
| --- | --- | --- | --- |
| 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 | Declare a new object called 'Kid' using a constructor function and give it the properties of our old object `kid`. Add a new property called “address”. | Constructor functions are created using largely the same syntax as other functions:  “function” method, followed by Constructor Class name, followed by parameters in parentheses. What is slightly different is how we pass arguments: use “this.property1” = property1 etc | let elf007 = `Buddy`;      let kid = {          elfManager : elf007,          fullName : [`Cindy`,`Lou`],          nice : true,          nocppy : 16, *// NoCPPY = "Number of Cookies Provided Previous Year"*      }      function Kid(elfManager, fullName, nice, nocppy, address) {          this.elfManager = elfManager;          this.fullName = fullName;          this.nice = nice;          this.nocppy = nocppy;          this.address = address;      }; |
| 2 | Declare a new instance object using your constructor function, called something like "kid1", assign our original “kid” values to it, and then log both the instance object, and the property fullName.  Note: for the "address" property, store an array of strings. | In order to create an instance, you need to declare a new variable as being equal to the original constructor object, and using the “new” method | let kid1 = new Kid(elf007, [`Cindy`,`Lou`], true, 16, [`5 Who Lane`,`Horton's Nose`, `Clover Pettal #5`])      console.log(kid1);      console.log(kid1.fullName); |
| 3 | Create three additional instances of your "Kid" object with new values - at least one of whom should be bad and have 0 cookies given last year |  | let kid2 = new Kid(elf007, [`Max`,`Grinch`], false, 0, [`Mount Crumpit`,`Horton's Nose`, `Clover Pettal #5`])      let kid3 = new Kid(elf007, [`Boo`,`Who`], true, 7, [`5 Who Lane`,`Horton's Nose`, `Clover Pettal #5`])      let kid4 = new Kid(elf007, [`Ozzy`,`Who`], true, 3, [`17 Who Lane`,`Horton's Nose`, `Clover Pettal #5`])      console.log(kid2);      console.log(kid3);      console.log(kid4); |
| 4 | Create a function that will store all kids who are "nice" in a new variable, using an ‘if’ statement. Note: make sure to store your instanced kids in a variable first | In addition to an array containing all of your previously-generated “kids” objects, you will need to declare a function that   * Declares an output variable (that will contain a new list) * Use a loop of your choice (say, a forEach loop) to go through each kid * A conditional to check whether the “nice” property is *true* for each kid (and stores them in your output list if-so), and * Return your new, filled-with-nice kids list   You may also want to turn off your older log to terminal instructions in order to clear things up for debugging this part | let kids = [kid1, kid2, kid3, kid4]      function naughtyNice(kidsArray) {  *// console.log(kidsArray);*              let output = [];              kidsArray.forEach(function(element) {  *// console.log(element)*                  if (element.nice) {                      output.push(element)                  }              });              return output      }      const finalList = naughtyNice(kids);      console.log(finalList); |
| 5 |  |  |  |
| 6 |  |  |  |
| ... |  |  |  |