|  |  |
| --- | --- |
| Sprint 9, Assignment 2.4 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” | “Accessing Nested Objects Parameters” |
| 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. | El capitan Kringle’s take-off is fast-approaching, but the head admin elf has just realized that some children may have more than one home or delivery address. You will need to add the ability to store multiple addresses (as nested objects) to your code. You will also need to make your code more readable and user-friendly. |
| The Assignment  A short specific description of the assignment and tasks using bullet points that the student will need to do. | * Create nested objects, * Access properties of those objects, and * Use conditional chaining to check for the existence of various properties |

|  |  |  |  |
| --- | --- | --- | --- |
| 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 | Create two nested objects with properties, and store them in one of your ‘kid’ instances | You need to make your address data more accessible by putting it in two nested objects. You currently have this data stored as multiple strings in an array. To begin, add two objects to one of your kid instances (or to a new, fifth instance if you'd rather). Next, create keys in these nested objects that we can associate with your address data: in addition to a new, special key called "type", you should use keys you often find in delivery forms - like "line 1", "line 2", "city", "state", and "zip".  Hint: objects have one or more key-value pairs wrapped by curly braces. You can store as many objects as you want in an array | let kid5 = new Kid(elf007, [`Mabel`,`Pines`], true, 76, [{          type : "Primary",          line1 : "Care of grunkle Stan Pines",          line2 : "618 Gopher Rd",          city : "Gravity Falls",          state : "OR",          zip : "97060"          },          {          type : "Secondary",          line1 : "Care of Mr. & Mrs. Pines",          line2 : "199 Oakland Ave.",          city : "Piedmont",          state : "CA",          zip : "94611"          }      ]) |
| 2 | Log the instance you created or changed to make sure that you can see your new, nested objects.  Also, see if you can log one or more specific properties of the first object. | Hint: access keys with the following syntax – [object’s name].[key of parent object].[key of child object] | console.log(kid5);      console.log(kid5.address[0].line1 + ", " + kid5.address[0].line2); |
| 3 | Convert your two nested objects from \*elements in an array\* to \*values associated with keys\*.  Also, log a specific object, and a property of that object. | This is your third re-structuring of the data. Your goal is to store your data in more and more easily-accessible forms. While storing your data as objects instead of strings makes getting specific pieces of it easier, you also want to be able to get the groups of data easily, too. Storing them based on, say, a categorical title will make accessing them even easier. Once finished, you should be able to call kid5.address.alternate.  Hint: you’re using the power of key-value pairs. Make your keys be “primary” and “alternate”, and store your address objects as values for them in the array. | let kid5 = new Kid(elf007, [`Mabel`,`Pines`], true, 76, {          primary : {          type : "Primary",          line1 : "Care of grunkle Stan Pines",          line2 : "618 Gopher Rd",          city : "Gravity Falls",          state : "OR",          zip : "97060"          },          alternate : {          type : "Secondary",          line1 : "Care of Mr. & Mrs. Pines",          line2 : "199 Oakland Ave.",          city : "Piedmont",          state : "CA",          zip : "94611"          }      })      console.log(kid2);      console.log(kid3);      console.log(kid4);      console.log(kid5.address.alternate); |
| 4 | Use conditional chaining to search whether a kid has a "secondary" address. Include an appropriate error message, in case the search is unsuccessful. | Hint: use “?” before “.”  Hint: use logical “or” pipes to give your program a choice between a) showing you what you’ve asked for and b) showing you an error message you’d find helpful  Bonus: can you create a conditional chain to check whether a kid was nice? | console.log(kid4?.address?.secondary || "Doesn't have secondary address")      console.log(kid2?.nice == true || "Wasn't good this year") |
| 5 |  |  |  |
| 6 |  |  |  |
| ... |  |  |  |