Session 4 Homework

Code For Everyone JavaScript

|  |  |
| --- | --- |
|  | **Study** |

1. for and Object

Run the following code, observe and then answer the questions

const product = {

name: ‘Xiaomi rice cooker’,

price: 1700,

brand: ‘Xiaomi’,

color: ‘white’

};

for (let x in product) {

console.log(x);

}

Questions:

* 1. What does x receives from product, property or value?

*A: Property*

* 1. Use the for loop to print/log out the following output

*A:*

const product = {

    name: "Xiaomi rice cooker",

    price: 1700,

    brand: "Xiaomi",

    color: "white"

};

*for* (let x in product) {

    console.log(x);

}

console.log(product);



1. Learn about destructuring object in the following tutorials:

[Object destructuring in ES6](https://dev.to/sarah_chima/object-destructuring-in-es6-3fm)

[ES6 destructuring: the complete guide](https://codeburst.io/es6-destructuring-the-complete-guide-7f842d08b98f)

Then use one line of code to destructure to obtain subject, dueDate and assignTo from this object:

const task = {

subject: ‘Implement login feature’,

createdBy: ‘Hoang Ngoc Duc’,

assignTo: ‘Nguyen Phuong Nam’,

dueDate: ‘2019-10-08T18:00:24+0000’,

expectedHours: 0.5,

};

let { subject, createdBy, assignTo, dueDate, expectedHours } = task;

console.log(subject);

console.log(dueDate);

console.log(assignTo);

1. [Here](https://gist.githubusercontent.com/qhuydtvt/6870e14e544455f6de6081a83e365b5b/raw/adb147e19259e3ee9b093cb71228026e2417ab09/jobs.js) is the actual data from a job search site, copy all of the data, assign it to a variable or a constant named jobSearch in your code. Log or print it out to see its structure then answer the following questions:
   1. What is the data type of the **outermost layer** (Number, String, Object, Array or else)?

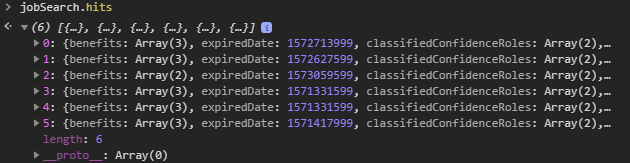
*Outermost layer is an Object*

console.log(typeof(jobSearch));

*"object"*

* 1. The hits property is where the job results are stored, is it a Number, String, Array, Object or something else?

*Hits property is an Array*



* 1. In each hit of hits, how to find the job’s title, locations, salary, benefits, skills and requirements

|  |  |
| --- | --- |
|  | **Review** |

1. Initialize an object to represent a dictionary, with properties as keyword and values as explanation, the initial values are from this table

|  |  |
| --- | --- |
| **Keyword** | **Explanation** |
| debug | The process of figuring out why your program has a certain error and how to fix it |
| done | When your task is complete, the only thing you have to do is to wait for users to use it (no additional codes or actions needed) |
| defect | The formal word for ‘error’ |
| pm | The short version of Project Manager, the person in charge of the final result of a project |
| ui/ux | UI means User Interface, UX mean User Experience, are the process to define how your products looks and feels |

let myDict = {

    debug: "The process of figuring out why your program has a certain error and how to fix it",

    done: "When your task is complete, the only thing you have to do is to wait for users to use it (no additional codes or actions needed)",

    defect: "The formal word for ‘error’",

    pm: "The short version of Project Manager, the person in charge of the final result of a project",

    "ui/ux": "UI means User Interface, UX mean User Experience, are the process to define how your products looks and feels"

}

* 1. Write a script to simulate the lookup of the dictionary initialized in the previous example

1. alert("Hi there, this is dev dictionary");
2. let traCuu = prompt("Enter a keyword");
3. *if* (typeof myDict[traCuu] === "undefined") {
4. alert (`We could not find your word: "${traCuu}"`)
5. } *else* {
6. alert(myDict[traCuu]);
7. }

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |

* 1. Update your script so that it can let users contribute the explanation to the dictionary

1. let tuKhoa = prompt("Enter a keyword");
2. *if* (typeof myDict[tuKhoa] === "undefined") {
3. const  newKeywordExplaination = prompt(`We could not find your word: "${tuKhoa}", leave your explanation`);
4. myDict[tuKhoa] = newKeywordExplaination;
5. alert("Done");
6. } *else* {
7. alert(myDict[tuKhoa]);
8. }

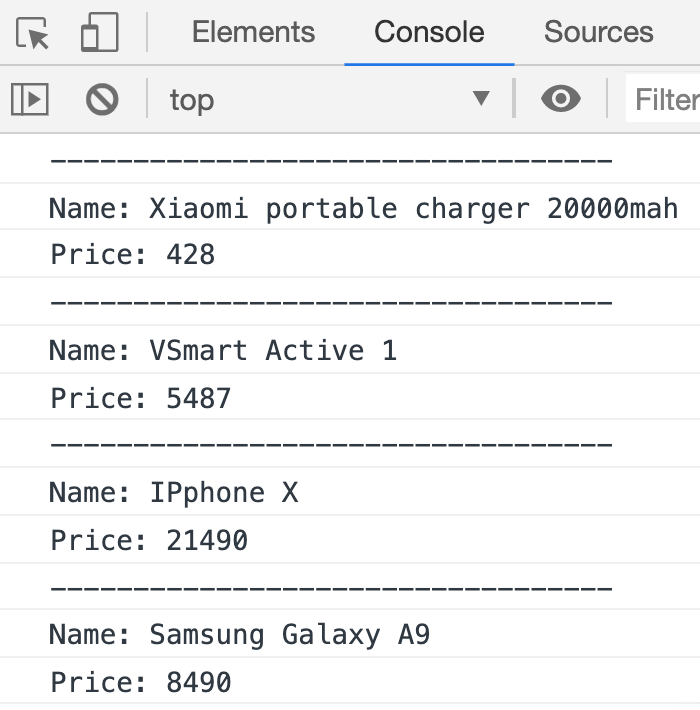
|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

*Note: If you want, you and handle the exceptions from user input, especially when users leave their explanation blank when the word is not found.*

1. Initialize a variable named products, containing an array of products, each product has a name, price, brand, category, and color

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **#** | **Name** | **Brand** | **Price** | **Color** | **Category** |
| 1 | Xiaomi portable charger 20000mah | Xiaomi | 428 | White | Charger |
| 2 | VSmart Active 1 | VSmart | 5487 | Black | Phone |
| 3 | IPhone X | Apple | 21490 | Gray | Phone |
| 4 | Samsung Galaxy A9 | Samsung | 8490 | Blue | Phone |

* 1. Print/log name and price of all the products out



* 1. Write a script printing/logging out the products with their number, then print/logging out the details of a product with its position entered by users

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |

* 1. Write a script printing/logging out the products based on category input by users

|  |  |  |
| --- | --- | --- |
|  |  |  |

|  |  |  |
| --- | --- | --- |
|  |  |  |

* 1. Add providers to each product

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **#** | **Name** | **Brand** | **Price** | **Color** | **Category** | **Providers** |
| 1 | Xiaomi portable charger 20000mah | Xiaomi | 428 | White | Charger | Phukienzero  Dientuccc |
| 2 | VSmart Active 1 | VSmart | 5487 | Black | Phone | Tgdd  Ddghn  VhStore |
| 3 | IPhone X | Apple | 21490 | Gray | Phone | Tgdd |
| 4 | Samsung Galaxy A9 | Samsung | 8490 | Blue | Phone | Tgdd |

And printing/logging out all of the products

|  |  |  |
| --- | --- | --- |
|  |  |  |

* 1. (Optional) Search the products based on the wanted provider entered by users, if you need more directions, find the hints at the end of this homework

|  |  |  |
| --- | --- | --- |
|  |  |  |

|  |  |
| --- | --- |
|  | **Serious exercices** |

1. Write a script to store and process the learning tasks to become a front-end developer
   1. Print it out

|  |  |  |
| --- | --- | --- |
|  |  |  |

* 1. Let users add new task

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |

* 1. Let users update task

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

* 1. Let users complete task

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |

* 1. Let users delete task (No illustrative image, you already know the drill)
  2. (Optional) Make printing / logging better

|  |  |  |
| --- | --- | --- |
|  | and |  |

|  |  |
| --- | --- |
|  | [**Turtle**](https://turtle.mindx.edu.vn/) **exercices** |

1. Given the object

const pos = {

x: 200,

y: 50,

}

Move the turtle to the provided position, use [penup()](https://github.com/raimohanska/turtle-js) and [pendown()](https://github.com/raimohanska/turtle-js) to NOT leave any traces

|  |  |  |
| --- | --- | --- |
| Actual result |  | Hint |

1. Given the object

const square = {

x: 100,

y: 50,

width: 20,

};

Use [JS Turtle](https://turtle.mindx.edu.vn/), to draw a square at position (x, y) with size width

|  |  |  |
| --- | --- | --- |
| Actual result |  | Hint |

1. Given the object

const rect = {

x: 100,

y: 50,

width: 20,

height: 40,

};

Use [JS Turtle](https://turtle.mindx.edu.vn/), to draw a rectangle at position (x, y) with size width and height

|  |  |  |
| --- | --- | --- |
| Actual result |  | Hint |

1. Given [this data structure](https://gist.githubusercontent.com/qhuydtvt/86ec0e962782b0d7509e881ad18c3447/raw/0add6fcf2ab007deef6d373aec3ed8ad4474cc58/turtleCmd.js), in which:

{

shape: 'rect',

x: 8,

y: 70,

width: 12,

height: 40,

}

rect means draw a rectangle with the respective position (x, y) and size (width, height)

{

shape: 'square',

x: 20,

y: 40,

width: 50,

},

square means draw a square with the respective position (x, y) and size (width)

Go through and execute all command.

*Note: To make turtle go to the initial position with initial angle, use* [*home()*](https://github.com/raimohanska/turtle-js) *statements.*

1. (Optional) Add circle to the command

const circle = {

shape: 'circle',

x: 100,

y: 50,

radius: 30,

};

|  |  |  |
| --- | --- | --- |
| Actual result |  | Hint |

1. (Optional) Execute all command from [this](https://gist.githubusercontent.com/qhuydtvt/1e63092d33122aded10e72b72337188c/raw/68c610d897555cdf186e7a0e105ed2e866caf313/cmdExtended.js)

|  |  |
| --- | --- |
|  | **Tools** |

1. Sometimes, you are given a very large object, which is hard to read, [this](https://pokeapi.co/api/v2/pokemon/psyduck) for example. It can be made much more readable if you using prettify / format tool, like [this Chrome Extension](https://chrome.google.com/webstore/detail/json-formatter/bcjindcccaagfpapjjmafapmmgkkhgoa?hl=en). Install it, reload the example and see the results. Learn how to switch between raw mode and parsed mode at the top right corner of the extension. Submit your screenshots to demonstrate your  usage

|  |  |  |
| --- | --- | --- |
|  |  |  |

1. Large objects are hard to read thus hard to analyze. To overcome this, you can log/print the object to the console, or you can use some online tools to analyze it better. [JSON Editor Online](https://jsoneditoronline.org/) is one of these. Learn how to use it (Just copy your data to the left panel of the Editor, press the ▶ button and see the result at the right panel). Submit your screenshots to demonstrate your understanding

|  |  |
| --- | --- |
|  | **Nice-to-do** |

1. (Optional) Get jobs data from this [link](https://gist.githubusercontent.com/qhuydtvt/6870e14e544455f6de6081a83e365b5b/raw/adb147e19259e3ee9b093cb71228026e2417ab09/jobs.js), copy the whole content and assign it a variable or a constant named jobData
   1. Get all the job hits

|  |  |  |
| --- | --- | --- |
|  |  |  |

* 1. Get the first job hit

|  |  |  |
| --- | --- | --- |
|  |  |  |

* 1. Get jobTitle of the first job

|  |  |  |
| --- | --- | --- |
|  |  |  |

* 1. Get the benefits of the first job hit

|  |  |  |
| --- | --- | --- |
|  |  |  |

* 1. Log out first job hit benefit values

|  |  |  |
| --- | --- | --- |
|  |  |  |

* 1. Log out jobTitle and benefitValue of all job hits

|  |  |  |
| --- | --- | --- |
|  |  |  |

* 1. Log out jobTitle, locations, skills, jobSalary of all job hits

|  |  |  |
| --- | --- | --- |
|  |  |  |

1. (Optional) There are at least two ways to delete a property-value pair from an object, the first one is the one you learned in class, to use delete keyword (which you already learned).. The second one is to create a new object without the property-value pair and just use the new object from then. For example:

const oldData = {

firedRice: {

Price: 30,

vnName: ‘Com rang dua bo’

},

**noddle: {**

**price: 20,**

**vnName: ‘My tom chanh’**

**},**

pho: {

price: 35,

vnName: ‘Pho bo tai chin’

},

};

If noddle removal is needed, a new object named newData is created containing data from oldData object, without noddle property. This gives the benefit of preserving the old data so it can be traced back when debugging in the future.

console.log(newData);

// Console

|  |  |  |
| --- | --- | --- |
|  |  |  |

**This can be done elegantly using JS 6 rest operator**, learn it and write an example to demonstrate your understanding. If you need hints, find them at the end of this homework

|  |  |
| --- | --- |
|  | **Hints** |

**Review**

5.5. After getting category from users, loop through all of the products, with each product, get respective providers and check whether the user-entered category is in the providers array (the quick way is to use Array indexOf function then check whether the result equals -1 or not), if so, print/log out the results