JavaScript –Assessment #2

Assigned 10/27/22 @ 3:40 PM AST

Due 10/27/22 @ 5:40 PM AST (time extension requested)

Completed 10/27/22 @ 4:45 PM AST (finished early!)

<https://lms.grandcircus.co/mod/assign/view.php?id=22743>

<https://docs.google.com/document/d/1o9SONuSoq3cwgPxc84Iu4cLwsbNQrPhGe-nG9BK4hlk/preview>

<https://github.com/gc-submissions/assessment-2-clintmsmith>

**Task:** Declare a number of functions that use loops and conditionals. Also, create an array of sandwiches and several more functions that work with a sandwich array.

**Submission:** Commit and push to submit.

**Build Specifications:**Create a file named sandwich.js. Complete all of the following tasks in that file. Each task is worth one point unless otherwise noted. NOTE: You may write additional code in sandwich.js if you like. It will not count against you. For example, it’s probably a good idea to call the functions you make to test them.

1. Declare a function named countdown
   * Parameter(s): n/a
   * Functionality: uses a loop to log, one by one, the numbers 5, 4, 3, 2, and 1
2. Declare a function named countUp
   * Parameter(s):  start, end
   * Functionality: uses a loop to log, one by one, the numbers from start through end, including both start and end.
3. Declare a function named isLong
   * Parameter(s):  word
   * Functionality: returns true if word is longer than 10 characters and false otherwise.
4. Declare a function named whatWaterState
   * Parameter(s):  temperature
   * Functionality: logs “solid” if temperature is 32 or less, “liquid” if temperature is above 32 and below 212, and “gas” if temperature is 212 or above. (Pay careful attention to < vs <= and > vs >= !)
5. Declare a variable named sandwiches that is initialized to an array with the following objects:

|  |  |  |  |
| --- | --- | --- | --- |
| **type** | **calories** | **cost** | **isVegetarian** |
| italian | 1000 | 7.69 | false |
| veggie | 500 | 5.50 | true |
| meatball | 900 | 8.25 | false |
| jackfruit | 650 | 8.50 | true |

1. [2 points] Declare a function named makeSandwich
   * Parameter(s): sandwichArray, type, calories, cost, isVegetarian
   * Functionality: construct an object and add it to the end of the provided array. Use the parameters for the values. The object should have the same properties as the objects already in the array.
2. [2 points] Declare a function named findVegetarianSandwich
   * Parameter(s): sandwichArray
   * Functionality: return the first object in sandwichArray that is vegetarian.
3. Declare a function named findCheapestSandwich
   * Parameter(s): sandwichArray
   * Functionality: return the object that has the lowest cost

Found the solution to this problem here:

<https://daily-dev-tips.com/posts/javascript-find-min-max-from-array-of-objects/>

**JavaScript:**

// Item 1 - countDown

function countDown () {

    for (let counter = 5; counter > 0; counter--)

    console.log(counter);

}

countDown();

// Item 2 - countUp

function countUp (start, end){

    for (let counter = start; counter <= end; counter++)

    console.log(counter);

}

countUp (3, 10);

countUp (1, 6);

// Testing various values for 'start' and 'end' arguments / parameters

// Item 3 - isLong

function isLong (word){

    if (word.length > 10){

        return true;

    } else {

        return false;

    }

}

console.log(isLong ("assess"));       // 6 characters

console.log(isLong ("assessment"));   // 10 characters

console.log(isLong ("assessments"));  // 11 characters

// Testing various lengths of words to be sure it works correctly

// Item 4 - whatWaterState

function whatWaterState (temperature){

    if (temperature <= 32){

        console.log('solid');

    } else if (temperature <= 211) {

        console.log('liquid');

    } else if (temperature >= 212) {

        console.log('gas');

    }

}

whatWaterState (31);

whatWaterState (32);

whatWaterState (33);

whatWaterState (210);

whatWaterState (211);

whatWaterState (212);

whatWaterState (213);

// Testing around the borders of each temperature threshold

// Item 5 - sandwiches array

let sandwiches = [

    {type: 'italian', calories: 1000, cost: 7.69, isVegetarian: false},

    {type: 'veggie', calories: 500, cost: 5.50, isVegetarian: true},

    {type: 'meatball', calories: 900, cost: 8.25, isVegetarian: false},

    {type: 'jackfruit', calories: 650, cost: 8.50, isVegetarian: true},

]

console.log(sandwiches);

// Item 6 - makeSandwich

function makeSandwich (sandwichArray, type, calories, cost, isVegetarian){

    let newSandwich = {

        type: type,

        calories: calories,

        cost: cost,

        isVegetarian : isVegetarian

    }

    sandwichArray.push(newSandwich);

    console.log(sandwichArray);

}

makeSandwich(sandwiches, 'turkey', 875, 7.75, false);

makeSandwich(sandwiches, 'pastrami', 975, 8.75, false);

// Tested adding multiple sandwiches

// Item 7 - findVegetarianSandwich

function findVegetarianSandwich (sandwichArray){

   let arrayResult = [];

   for (let i = 0; i <= sandwichArray.length - 1; i++){

    if(sandwichArray[i].isVegetarian === true){

        arrayResult.push(sandwichArray[i]);

        break;

    }

   }

   return arrayResult;

}

console.log(findVegetarianSandwich(sandwiches));

// Item 8 - findCheapestSandwich

function findCheapestSandwich (sandwichArray){

    let lowest = sandwichArray.reduce((cost1, cost2) => {

        return cost1.cost <= cost2.cost ? cost1 : cost2;

    });

    console.log(lowest);

}

findCheapestSandwich(sandwiches);

// To test this I changed the 'cost' value in the object array above.  Started with 'veggie', then increased the price above 'italian' and got that as the result,

// and did the same thing until I got 'meatball' and ultimately 'jackfruit'.  It the cheapest is the same cost as another sandwich, it returns the first option.

**Terminal Output:**

***Item 1 – Count from 5 down to 1***

5

4

3

2

1

***Item 2 – Count from ‘start’ to ‘end’ based on arguments entered***

3

4

5

6

7

8

9

10

*testing Item 2 with different ‘start’ and ‘end’ values*

1

2

3

4

5

6

***Item 3 – testing if parameter / argument ‘word’ is longer than 10 letters***

false – used ‘assess’ (6)

false – used ‘assessment’ (10)

true – used ‘assessment’ (11)

***Item 4 – state of water: solid ( <32 ), liquid ( 33 – 211 ), or gas ( >= 212 )***

solid – used 31

solid – used 32

liquid – used 33

liquid – used 210

liquid – used 211

gas – used 212

gas – used 213

***Item 5 – create an array of sandwich objects***

[

{ type: 'italian', calories: 1000, cost: 7.69, isVegetarian: false },

{ type: 'veggie', calories: 500, cost: 5.5, isVegetarian: true },

{ type: 'meatball', calories: 900, cost: 8.25, isVegetarian: false },

{ type: 'jackfruit', calories: 650, cost: 8.5, isVegetarian: true }

]

***Item 6 – make a new sandwich and add it to the original array above (added two)***

[

{ type: 'italian', calories: 1000, cost: 7.69, isVegetarian: false },

{ type: 'veggie', calories: 500, cost: 5.5, isVegetarian: true },

{ type: 'meatball', calories: 900, cost: 8.25, isVegetarian: false },

{ type: 'jackfruit', calories: 650, cost: 8.5, isVegetarian: true },

**{ type: 'turkey', calories: 875, cost: 7.75, isVegetarian: false }**

]

[

{ type: 'italian', calories: 1000, cost: 7.69, isVegetarian: false },

{ type: 'veggie', calories: 500, cost: 5.5, isVegetarian: true },

{ type: 'meatball', calories: 900, cost: 8.25, isVegetarian: false },

{ type: 'jackfruit', calories: 650, cost: 8.5, isVegetarian: true },

{ type: 'turkey', calories: 875, cost: 7.75, isVegetarian: false },

**{ type: 'pastrami', calories: 975, cost: 8.75, isVegetarian: false }**

]

***Item 7 – return the first object in the array that is vegetarian***

[ { type: 'veggie', calories: 500, cost: 5.5, isVegetarian: true } ]

***Item 8 – return the object with the lowest cost***

{ type: 'veggie', calories: 500, cost: 5.5, isVegetarian: true }