# Lab: This

Problems for the in-class lab for the ["JavaScript Applications" course @ SoftUni](https://softuni.bg/courses/js-applications)

Submit your solutions in the SoftUni judge system at [https://judge.softuni.bg/Contests/Practice/Index/1854 - 0](https://judge.softuni.bg/Contests/Practice/Index/1854#0)

## Area and Volume Calculator

Write a function which **calculates** the **area** and the **volume** of a figure, which is **defined** by its **coordinates**   
(**x**, **y**, **z**).

|  |
| --- |
| area() |
| function area() {      return this.x \* this.y;  }; |

|  |
| --- |
| vol() |
| function vol() {      return this.x \* this.y \* this.z;  }; |

|  |
| --- |
| solve() |
| function solve(area, vol, input) {      //ToDo....  } |

### Input

You will receive **3** parameters - the **functions** **area** and **vol** and a **string**, which contains the figures' coordinates.

**For more information check the examples.**

### Output

The output should be **returned** as an **array** **of objects**. Each object has **two** **properties**: the figure's **area** and **volume**.

**[**

**{ area: ${area1}, volume: ${volume1} },**

**{ area: ${area2}, volume: ${volume2} },**

**. . .**

**]**

### Note:

**Submit only the solve function.**

### Examples

|  |  |
| --- | --- |
| Sample Input | Output |
| area, vol, '[  {"x":"1","y":"2","z":"10"},  {"x":"7","y":"7","z":"10"},  {"x":"5","y":"2","z":"10"}  ]' | [  { area: 2, volume: 20 },  { area: 49, volume: 490 },  { area: 10, volume: 100 }  ] |
| area, vol, '[  {"x":"10","y":"-22","z":"10"},  {"x":"47","y":"7","z":"-5"},  {"x":"55","y":"8","z":"0"},  {"x":"100","y":"100","z":"100"},  {"x":"55","y":"80","z":"250"}  ]' | [  { area: 220, volume: 2200 },  { area: 329, volume: 1645 },  { area: 440, volume: 0 },  { area: 10000, volume: 1000000 },  { area: 4400, volume: 1100000 }  ] |

## Person

Write a JS program which takes **first** & **last** names as **parameters** and returns an object with **firstName**, **lastName** and **fullName** ( **"{firstName} {lastName}"** ) properties which should be all **accessibles**, we discovered that "accessible" also means "mutable". This means that:

* If .**firstName** or .**lastName** have changed, then .**fullName** should also be changed.
* If .**fullName** is changed, then .**firstName** and .**lastName** should also be changed.
* If **fullName** is **invalid**, you should not change the other properties. A **valid** **full name** is in the format

**"{firstName} {lastName}"**

Note: For more information check the examples below.

### Examples

|  |
| --- |
| Sample Input |
| let person = new Person("Peter", "Ivanov");  console.log(person.fullName);*//Peter Ivanov*  person.firstName = "George";  console.log(person.fullName);//George Ivanov  person.lastName = "Peterson";  console.log(person.fullName);//George Peterson  person.fullName = "Nikola Tesla";  console.log(person.firstName)//Nikola  console.log(person.lastName)//Tesla |
| let person = new Person("Albert", "Simpson");  console.log(person.fullName);//Albert Simpson  person.firstName = "Simon";  console.log(person.fullName);//Simon Simpson  person.fullName = "Peter";  console.log(person.firstName) // Simon  console.log(person.lastName) // Simpson |

## ArrayMap

Write a function that takes **2** **parameters** (**array** and a **function**) that uses **.reduce()** to implement a simple version of **.map().**

### Input

You will receive **2** parameters - an **array**, and a **function**.

### Output

The output should be **returned** as a **new** **array** (changed according to the given function).

**For more information check the examples below.**

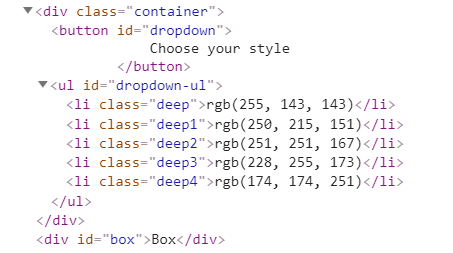
### Examples

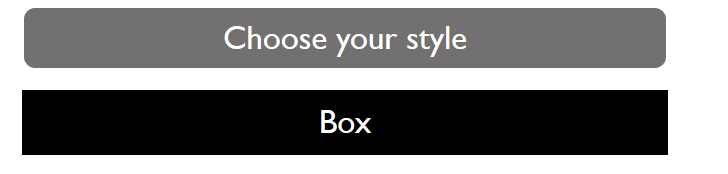
|  |
| --- |
| Sample exectuion |
| let nums = [1,2,3,4,5];  console.log(arrayMap(nums,(item)=> item \* 2)); // [ 2, 4, 6, 8, 10 ] |
| let letters = ["a","b","c"];  console.log(arrayMap(letters,(l)=>l.toLocaleUpperCase())) // [ 'A', 'B', 'C' ] |

## Dropdown Menu

### Use the Given Skeleton to Solve This Problem.

**Note: You Have NO Permission to Change Directly the Given HTML (Index.html File).**



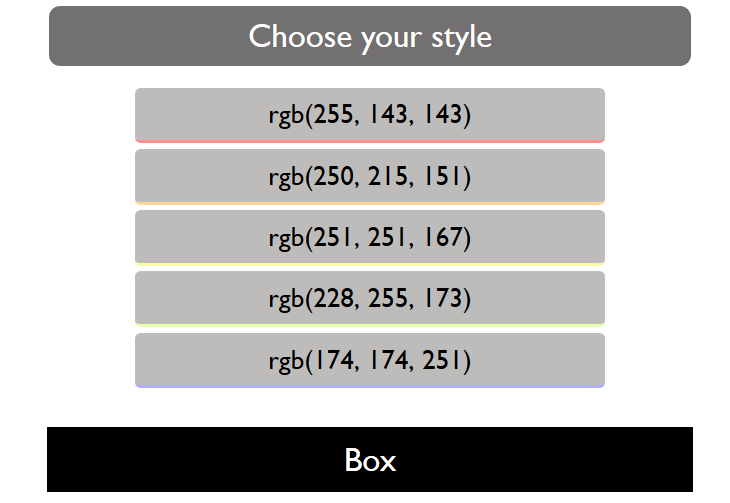


### Your Task

Write the missing JavaScript code to make the **Dropdown Menu** application work as expected.

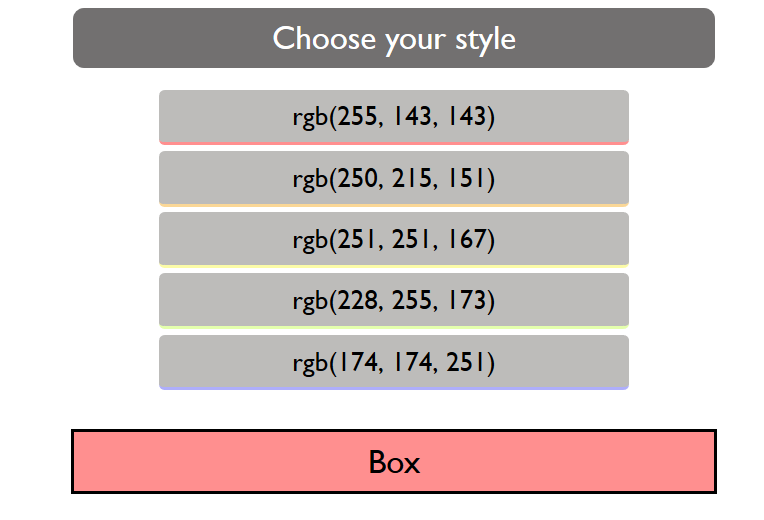
When you **click** on the [Choose your style] button, the elements of the menu should become visible.

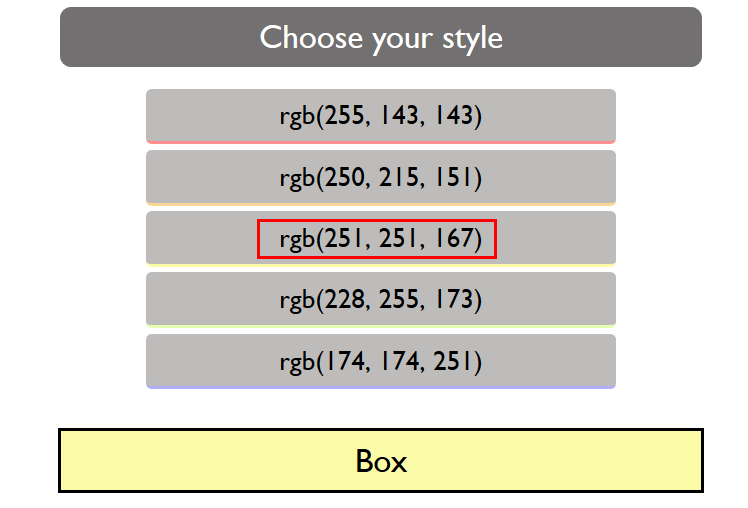




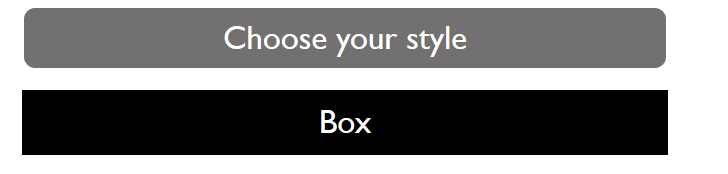
When you click on one of the items the background color of the box below should be changed to the RGB, which is displayed in the list item







When the **[Choose your style]** button **is clicked** again, you should hide the list items, and the box should be returned to its initial state.



## Spy

Write a function that takes **2** **parameters** **target**(an object) and **method**(a string) and tracks **how** **many** **times** the method of an object is **called**.

### Input

* **target**: an **object** containing the **method**
* **method**: a **string** with the **name of the method** on target to spy on

### Output

The output should be **returned** as an **object** with property **count**, which holds how many times the provided method is invoked.

### Examples

|  |
| --- |
| Sample exectuion |
| let obj = {  method:()=>"invoked"  }  let spy = Spy(obj,"method");  obj.method();  obj.method();  obj.method();  console.log(spy) // { count: 3 } |
| let spy = Spy(console,"log");  console.log(spy); // { count: 1 }  console.log(spy); // { count: 2 }  console.log(spy); // { count: 3 } |

### Hints

Check the code below.

