# Exercise: Function Context

Problems for in-class lab for the [“JavaScript Advanced” course @ SoftUni](https://softuni.bg/courses/javascript-advanced).

Submit your solutions in the SoftUni judge system at <https://judge.softuni.bg/Contests/1855/Exercise-Function-Context>.

## Company

**class** Company {  
 *//* ***TODO: implement this class...***  
}

### Your Task

### Write a Company Class, Which Supports the Described Functionality Below.

### Functionality

**Constructor()**

Should have these **1** property:

* **departments** - empty array

**AddEmployee({username}, {Salary}, {Position}, {Department})**

This function should add a new employee to the department with the given name.

* If one of the passed parameters is empty string (""), undefined or null, this function should throw an error with the following message:

"**Invalid input!"**

* If salary is less than 0, this function should throw an error with the following message:

**" Invalid input!"**

* If the new employee is hired successfully, you should add him into the departments array and return the following message:

**"**

**bestDepartment()**

This **function** should print the department with the highest average salary and its employees sorted by their salary by descending and by name in the following format:

**"Best department is: {best department's name}**

**Average salary: {best department's average salary}**

**{employee1} {salary} {position}**

**{employee2} {salary} {position}**

**{employee3} {salary} {position}**

**. . ."**

### Submission

Submit only your **Company class.**

### Examples

This is an example how the code is **intended to be used**:

|  |
| --- |
| Sample code usage |
| let c = new Company();  c.addEmployee("Stanimir", 2000, "engineer", "Construction");  c.addEmployee("Pesho", 1500, "electrical engineer", "Construction");  c.addEmployee("Slavi", 500, "dyer", "Construction");  c.addEmployee("Stan", 2000, "architect", "Construction");  c.addEmployee("Stanimir", 1200, "digital marketing manager", "Marketing");  c.addEmployee("Pesho", 1000, "graphical designer", "Marketing");  c.addEmployee("Gosho", 1350, "HR", "Human resources");  console.log(c.bestDepartment()); |
| Corresponding output |
| Best Department is: Construction  Average salary: 1500.00  Stan 2000 architect  Stanimir 2000 engineer  Pesho 1500 electrical engineer  Slavi 500 dyer |

## Fibonacci

Write a JS function that when called, returns the next Fibonacci number, starting at 0, 1. Use a **closure** to keep the current number.

### Input

There will be no input.

### Output

The **output** must be a Fibonacci number and must be **returned** from the function.

### Examples

|  |
| --- |
| Sample exectuion |
| let fib = getFibonator();  console.log(fib()); *// 1*  console.log(fib()); *// 1*  console.log(fib()); *// 2*  console.log(fib()); *// 3*  console.log(fib()); *// 5*  console.log(fib()); *// 8*  console.log(fib()); *// 13* |

## HEX

**class** Hex {  
 *//* ***TODO: implement this class...***  
}

### Your Task

### Write a Hex Class, Which Supports the Described Functionality Below.

### Functionality

**Constructor({value})**

Should have these **1** property:

* **value** - number

#### ValueOf()

#### This Function Should Return the Value Property of the Hex Class.

**ToString()**

This **function** will show its hexidecimal value starting with "0x"

**Plus({number})**

This function should add a number or Hex object and return a new Hex object.

**Minus({number})**

This function should subtract a number or Hex object and return a new Hex object.

**Parse({string})**

Create a parse class method that can **parse** Hexidecimal numbers and convert them to standard decimal numbers.

### Submission

Submit only your **Hex class.**

### Examples

This is an example how the code is **intended to be used**:

|  |
| --- |
| Sample exectuion |
| let FF = new Hex(255);  console.log(FF.toString());  FF.valueOf() + 1 == 256;  let a = new Hex(10);  let b = new Hex(5);  console.log(a.plus(b).toString());  console.log(a.plus(b).toString()==='0xF'); |
| **0xFF**  **0xF**  **true** |

## Table

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

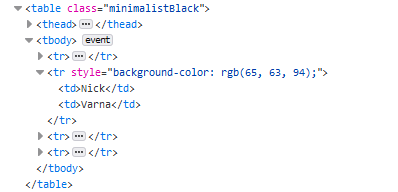
**Note: You Have NO Permission to Change Directly the Given HTML (Index.html File).**

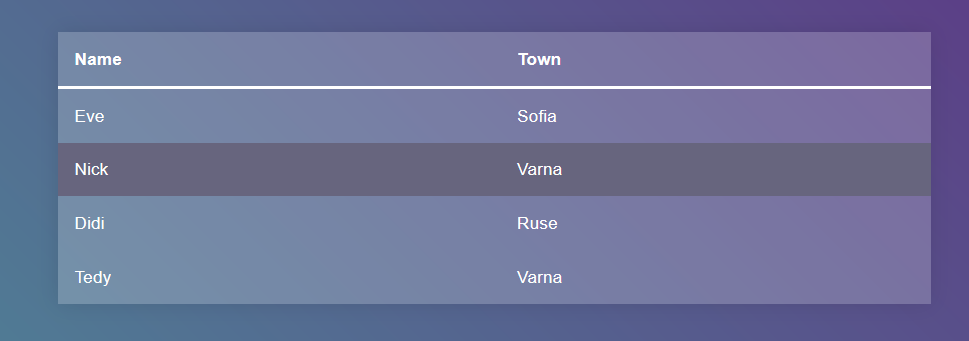


### Your Task

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

When you **click** on an item from the table you should change its **background** **color** to "#413f5e".

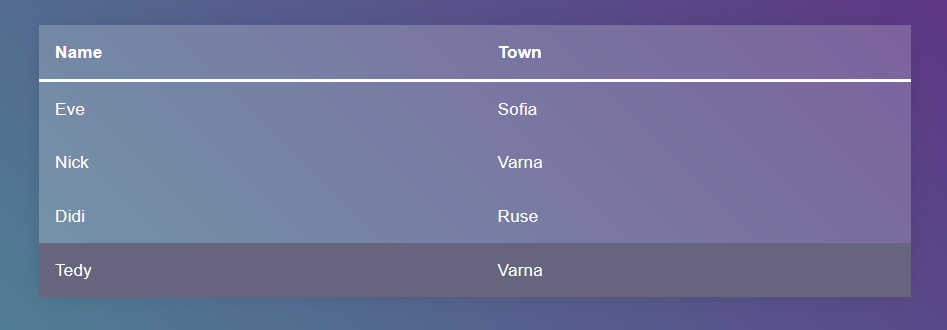


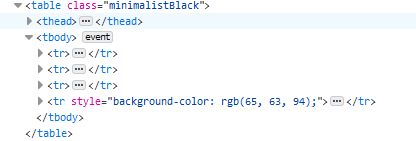


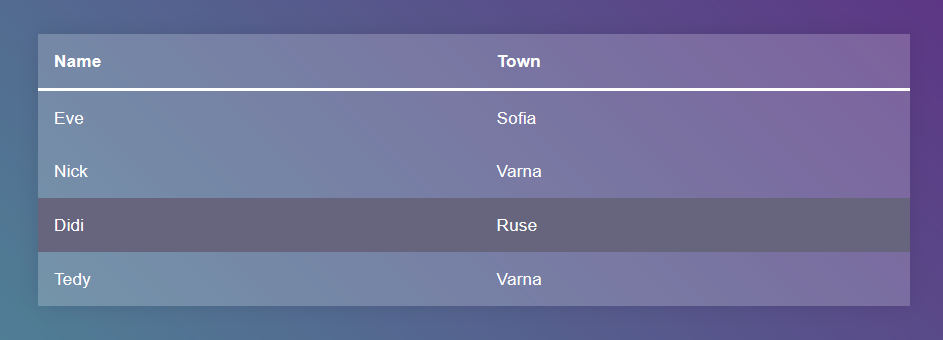
If the item you've clicked **already** **has** a **style** property you should **remove** it.

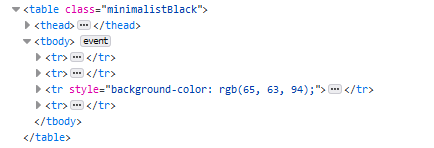


If one of the elements is **clicked** and you click **another** the first element's style property should be **removed** and you should **change** the **background** **color** of the **newly** **clicked** item.









Note: You **shouldn't** change the head of the table, even if it is clicked.

## Next Article

Write a JS program that sequentially **displays articles** on a web page when the user **clicks** a button. You will receive an **array of strings** that will initialize the program. You need to return a function that keeps the initial array in its closure and every time it’s called, it takes the first element from the array and displays it on the web page, inside a div with ID "**content**". If there are no more elements left, your function should do nothing.

### HTML and JavaScript Code

You are given the following **HTML** code:

|  |
| --- |
| article.html |
| <!DOCTYPE **html**> <**html lang="en"**> <**head**>  <**meta charset="UTF-8"**>  <**title**>Next Article</**title**>  <**style**>**div**{**width**:600**px**; **text-align**: **center**; **font-size**: 1.5**em**} **article**{**border**: 2**px solid blue**; **padding**: 2**em**; **margin**: 1**em**}</**style**>  <**script src="https://code.jquery.com/jquery-3.1.1.min.js" integrity="sha256-hVVnYaiADRTO2PzUGmuLJr8BLUSjGIZsDYGmIJLv2b8=" crossorigin="anonymous"**></**script**>  <**script src="next-article.js"**></**script**> </**head**> <**body**> <**div id="content"**></**div**> <**div**><**button onclick="*showNext***()**"**>Show Next Article</**button**></**div**> <**script**>  let ***articles*** =[  **"Cats are the most popular pet in the United States: There are 88 million pet cats and 74 million dogs."**,  **"A group of cats is called a clowder."**,  **"Cats have over 20 muscles that control their ears."**,  **"A cat has been mayor of Talkeetna, Alaska, for 15 years. His name is Stubbs."**,  **"The world's largest cat measured 48.5 inches long."** ];  let ***showNext*** = *getArticleGenerator*(***articles***); </**script**> </**body**> </**html**> |

It comes together with the following **JavaScript** code:

|  |
| --- |
| next-article.js |
| **function** *getArticleGenerator*(articles) {  *//* ***TODO*** } |

Your function will be called automatically, there is **no need** to attach any event listeners.

### Input

You will receive and **array** of strings.

### Output

Return a **function** that displays the array elements on the web page.

### Examples