**JS Advanced: Retake Exam 5 September 2017**

Problems for exam preparation 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/756/>.

**Problem 3. Repository (Simple Class)**

Write a JavaScript class **Repository** that has **props** (object that will validate an entity) and **data** (a Map which stores entities). All entities inside the repository have the **same properties** (listed in the props) and a **unique ID**, that is assigned when they are added, **starting at zero**.

|  |
| --- |
| **class** Repository {  *//* ***TODO: implement this class*** } |

The class **constructor** should receive one parameter – **props** (object), and initialize the **data** with a new **Map** instance**.** Implement the following features:

* Property **data** – **Map** that holds added entities
* Function **add(entity)** – adds an entity to the data; if successful, returns the resulting ID
* Function **get(id)** – returns the entity with given ID
* Function **update(id, newEntity)** – replaces the entity with the given id with the new entity
* Function **del(id)** – deletes an entity by given id
* Getter **count** – returns the number of stored entities

The **props** parameter is used to **validate** entities added to the repository and is an object with format:

**{**

**propName1: propType1,**

**propName2: propType2,**

**…**

**propNameN: propTypeN**

**}**

When an entity is **added** to the repository, it should be **validated** against the props object – it needs to have all of the properties that the props object has and their values must be of the specified type. For example, if **props** has a property **"name"** with value **"string"**, all entities added must have a **name** property with value of type **string**. If **any** property is **missing**, you should **throw** an **Error** with message: **"Property {propName} is missing from the entity!"**. If the property is present, but is of **incorrect** type, **throw** a **TypeError** with message **"Property {propertName} is of incorrect type!".** If validation is successful, add the entity to the repository with a new ID. Store entities in a Map where the key is the ID and the value is the entity.

To **update** an entity, we receive its **id** and the **new** entity object. If the id does **not** exist in the **data** throw an **Error** with message **"Entity with id: {id} does not exist!"**. Validate the **new** entity with the **same** validations and **replace** the old one with the new one.

To **delete** an entity, we receive only its **id**. If the id does **not** exist in the **data** throw an **Error** with message **"Entity with id: {id} does not exist!"**. After that **remove** the entity from the **map**.

**Examples**

This is an example how the **Repository** class is **intended to be used**. Make sure to comment out the parts that throw an error!

|  |
| --- |
| **Sample code usage** |
| *// Initialize props object* **let *properties*** = {  **name**: **"string"**,  **age**: **"number"**,  **birthday**: **"object"** }; *//Initialize the repository* **let *repository*** = **new *Repository***(***properties***); *// Add two entities* **let *entity*** = {  **name**: **"Kiril"**,  **age**: 19,  **birthday**: **new** Date(1998, 0, 7) }; ***repository***.add(***entity***); *// Returns 0* ***repository***.add(***entity***); *// Returns 1* **console**.log(***repository***.get(0));  *// {"name":"Kiril","age":19,"birthday":"1998-01-06T22:00:00.000Z"}*  **console**.log(***repository***.get(1));  *// {"name":"Kiril","age":19,"birthday":"1998-01-06T22:00:00.000Z"} //Update an entity* ***entity*** = {  **name**: **'Valio'**,  **age**: 19,  **birthday**: **new** Date(1998, 0, 7) }; ***repository***.update(1, ***entity***); **console**.log(***repository***.get(1));  *// {"name":"Valio","age":19,"birthday":"1998-01-06T22:00:00.000Z"} // Delete an entity* ***repository***.del(0); **console**.log(***repository***.count); *// Returns 1*  **let *anotherEntity*** = {  **name1**: **'Nakov'**,  **age**: 26,  **birthday**: **new** Date(1991, 0, 21) }; ***repository***.add(***anotherEntity***); *// should throw an Error* ***anotherEntity*** = {  **name**: **'Nakov'**,  **age**: 26,  **birthday**: 1991 }; ***repository***.add(***anotherEntity***); *// should throw a TypeError* ***repository***.del(-1); *// should throw Error for invalid id* |

We add **two** entities which are exactly the same. After that the second one is **updated** with a **different** name and lastly, we **delete** the entity with id **zero**. The corresponding output is **without** the errors.

**Constraints**

* The ID should change **only** when we **add** a new entity.
* The ID is unique per repository – if two repositories are instantiated, each has its own counter.

**Submission**

Submit **only** your class **Repository**.

**Hints**

Use **typeof** of determine the type of a property.