# JS Advanced: Retake Exam 05 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/699/>.

# 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).**

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

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

* Property props
* Function **addEntity(entity)** – adds an entity to the data.
* Function **updateEntity(id, newEntity)** – replaces the entity with the given id with the new entity.
* Function **deleteEntity(id)** – deletes an entity by given id.
* Function toString() – returns a string representation of **all** entities in the following format:

**{entityId}: {the entity object in string format}**

The **props object** is used to **validate** each entity we receive. For example an entity **should** have a property “**name**” which should be a “**string**”, property “**age**” which should be a “**number**”. The props object comes in format **{propName: propType, ...}.** Each **entity** should have an **id** which is **unique** and **auto incremented** every time we **add** one in the data map. The **map** stores values in format **{entityId: entityObj}.** The **id** should start at **zero** on repository **initialization**.

Before an entity is **added** to the data map it should be **validated** with the props object. If **any** property from the entity is **missing** eg. -> props object has a property “name”, but the entity object does not, you should **throw** an **Error** with the following message: “**Property {propertyName} is missing from the entity!**”. If the property is present, but is of **incorrect** type eg. -> property “name” should be a “string” but the entity value is an “object”, you should **throw** a **TypeError** with the following message: “**Property {propertName} is of incorrect type!**”**.** After all of these **validations**, **add** the entity to the data map and **increment** the id.

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 the following 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 the following 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**:

|  |
| --- |
| 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***.addEntity(***entity***); ***repository***.addEntity(***entity***); **console**.log(***repository***.toString()); *//Update an entity* ***entity*** = {  **name**: **'Valio'**,  **age**: 19,  **birthday**: **new** Date(1998, 0, 7) }; ***repository***.updateEntity(1, ***entity***); **console**.log(***repository***.toString()); *// Delete an entity* ***repository***.deleteEntity(0); **console**.log(***repository***.toString()); **let *anotherEntity*** = {  **name1**: **'Nakov'**,  **age**: 26,  **birthday**: **new** Date(1991, 0, 21) }; ***repository***.addEntity(***anotherEntity***); *// should throw an Error* ***anotherEntity*** = {  **name**: **'Nakov'**,  **age**: 26,  **birthday**: 1991 }; ***repository***.addEntity(***anotherEntity***); *// should throw a TypeError* ***repository***.deleteEntity(-1); *// should throw Error for invalid id* |
| Corresponding output |
| // **Add two entities**  0: {"name":"Kiril","age":19,"birthday":"1998-01-06T22:00:00.000Z"}  1: {"name":"Kiril","age":19,"birthday":"1998-01-06T22:00:00.000Z"}  // **Update the second**  0: {"name":"Kiril","age":19,"birthday":"1998-01-06T22:00:00.000Z"}  1: {"name":"Valio","age":19,"birthday":"1998-01-06T22:00:00.000Z"}  // **Delete the first**  1: {"name":"Valio","age":19,"birthday":"1998-01-06T22:00:00.000Z"} |

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.

### Submission

Submit **only** your class Repository.