# JS Advanced - Exam: 08.04.2020

Exam problems for the [“JavaScript Advanced” course @ SoftUni](https://softuni.bg/courses/javascript-advanced).

## Problem 2. Repository

Use the provided Repository **class** to solve this problem.

### Your Task

Using **Mocha** and **Chai** write **JS Unit Tests** to test the entire functionality of the Repository **class**. Make sure instances of it have all the required functionality and validation. You may use the following code as a template:

|  |
| --- |
| describe(**"*Tests* …"**, **function**() {  describe(**"*TODO* …"**, **function**() {  ***it***(**"*TODO …*"**, **function**() {  *//* ***TODO:*** …  });  });  *//* ***TODO:*** …  }); |

### Functionality

solution.js defines a **class** that contains information about a **Repository class**. An **instance** of the class should support the following operations:

* **Instantiation** with **one** **parameter** - The **props** parameter, which is used to **validate** entities added to the repository and is an object**,** and an **additional properties** called data ( **Map** that holds added entities).
* Getter count – returns the number of stored entities
* Function add(entity) – adds an entity to the data; if successful Store entities in a Map where the key is the ID and the value is the entity and returns the resulting ID. Before 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. 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 {propertyName} is of incorrect type!**"**.**
* Function getId(id) – returns the entity with given ID
* Function update(id, newEntity) – replaces the entity with the given id with the new entity. 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.
* Function del(id) – deletes an entity by given id. If the id does **not** exist in the **data** throw an **Error** with message "**Entity with id: {id} does not exist!**".

### Examples

|  |
| --- |
| Sample Code Usage |
| let properties = {      name: "string",      age: "number",      birthday: "object"  };  let repository = new Repository(properties);  let entity = {      name: "Pesho",      age: 22,      birthday: new Date(1998, 0, 7)  };  repository.add(entity);  repository.add(entity);  console.log(repository.getId(0));  console.log(repository.getId(1));  entity = {      name: 'Gosho',      age: 22,      birthday: new Date(1998, 0, 7)  };  repository.update(1, entity);  console.log(repository.getId(1));  repository.del(0);  console.log(repository.count); |
| Corresponding Output |
| **{ name: 'Pesho', age: 22, birthday: 1998-01-06T22:00:00.000Z }**  **{ name: 'Pesho', age: 22, birthday: 1998-01-06T22:00:00.000Z }**  **{ name: 'Gosho', age: 22, birthday: 1998-01-06T22:00:00.000Z }**  **1** |

### Submission

Submit your tests inside a describe() statement, as shown above.

let { Repository } = require("./solution.js");

const { assert } = require('chai');

describe("Initialize", function () {

    const properties = {

        name: 'string',

        age: 'number',

        birthday: 'object',

    };

    // valid

    const entity1 = {

        name: 'Gosho',

        age: 22,

        birthday: new Date(1998, 0, 7),

    };

    // valid

    const entity2 = {

        name: 'Peter',

        age: 37,

        birthday: new Date(1984, 3, 17),

    };

    // invalid

    const entity4 = {

        name1: 'Stamat',

        age: 29,

        birthday: new Date(1991, 0, 21),

    };

    // invalid

    const entity5 = {

        name: 'Stamat',

        age: '29',

        birthday: new Date(1991, 0, 21),

    };

    const entity6 = {

        name: 'Stamat',

        age: 29,

        birthday: 1991

    };

    it("Instantiation", function () {

        const repository = new Repository(properties);

        assert.equal(repository.props, properties);

        assert.equal((new Repository().props), undefined);

        assert.equal((repository.props.name), 'string');

        assert.equal((repository.props.age), 'number');

        assert.equal((repository.props.birthday), 'object');

        assert.deepEqual((repository.data), new Map());

        assert.deepEqual((repository.data.size), 0);

        assert.equal((repository.count), 0);

        assert.equal((repository.nextId()), 0)

        assert.equal((repository.nextId()), 1)

        assert.deepEqual(repository.data, new Map());

        assert.equal((typeof repository.nextId), 'function');

    });

    it("add", function () {

        const repo = new Repository(properties);

        assert.equal(repo.add(entity1), 0);

        assert.equal((repo.data.size), 1);

        assert.equal(repo.add(entity2), 1);

        assert.equal((repo.data.size), 2);

        assert.equal(repo.nextId(), 2);

        assert.equal(repo.data.get(0), entity1);

        assert.equal(repo.data.get(1), entity2);

        assert.throw(() => repo.add(entity4), `Property name is missing from the entity!`);

        assert.throw(() => repo.add(entity5), `Property age is not of correct type!`);

        assert.throw(() => repo.add(entity6), `Property birthday is not of correct type!`);

    });

    it("addThrowErrors", function () {

        const repo = new Repository(properties);

        assert.throw(() => repo.add(entity4), `Property name is missing from the entity!`);

        assert.throw(() => repo.add(entity5), `Property age is not of correct type!`);

        assert.throw(() => repo.add(entity6), `Property birthday is not of correct type!`);

    });

    it("update", function () {

        const repo = new Repository(properties);

        repo.add(entity1);

        repo.update(0, entity2);

        assert.equal(repo.getId(0), entity2);

    });

    it("updateThrowErrors", () => {

        const repo = new Repository(properties);

        repo.add(entity1);

        repo.update(0, entity2);

        assert.throw(() => repo.update(3, entity2), `Entity with id: 3 does not exist!`)

        assert.throw(() => repo.update(-1, entity2), `Entity with id: -1 does not exist!`)

        assert.throw(() => repo.update(0, entity4), 'Property name is missing from the entity!')

        assert.throw(() => repo.update(0, entity5), 'Property age is not of correct type!')

        assert.throw(() => repo.update(0, entity6), 'Property birthday is not of correct type!')

    });

    it('get id', () => {

        const repo = new Repository(properties);

        repo.add(entity1);

        assert.equal(repo.getId(0), entity1);

        assert.throw(() => repo.getId(5), 'Entity with id: 5 does not exist!');

    })

    it('del', () => {

        // testing the method 'del' with valid and invalid entity

        const repo = new Repository(properties);

        repo.add(entity1);

        assert.throw(() => repo.del(9)), `Entity with id: 9 does not exist!`;

        repo.del(0);

        assert.equal((repo.data.size), 0);

        assert.throw(() => repo.del(0)), `Entity with id: 0 does not exist!`;

        assert.equal((repo.data.size), 0);

    });

    it('getCount', () => {

        const repo = new Repository(properties);

        repo.add(entity1);

        assert.equal((repo.data.size), 1)

    });

});