**JS Advanced - Exam**

**Problem 3. Unit Testing**

**Your Task**

Using **Mocha** and **Chai** write **JS Unit Tests** to test a variable named **pizzUni**, which represents an object. You may use the following code as a template:

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

The object that should have the following functionality:

* **makeAnOrder(obj)** - A function that accepts an object:
  + the object includes {orderedPizza: ‘the name of the pizza’, orderedDrink: ‘the name of the drink’}
  + the function checks if there are ordered pizza and a drink.
  + Then the function returns confirmation message for your order
* **getRemainingWork(statusArr)** - A function that accepts array:
  + the array should look like: [{pizzaName: ‘the name of the pizza’, status: ‘ready’ / ‘preparing’ }, …]
  + the function checks the status of the order and returns a message with the order status
* **orderType(totalSum, typeOfOrder)** - A function that accepts two parameters (number, string):
  + the function first checks what is the type of the order (‘Carry Out’ , ‘Delivery’)
  + if the type of the order is ‘Carry Out’ you get 10% discount
  + then the function returns the total sum of the order

**JS Code**

To ease you in the process, you are provided with an implementation which meets all of the specification requirements for the **pizzUni** object:

|  |
| --- |
| **pizza.js** |
| let pizzUni = {      makeAnOrder: function (obj) {          if (!obj.orderedPizza) {              throw new Error('You must order at least 1 Pizza to finish the order.');          } else {              let result = `You just ordered ${obj.orderedPizza}`              if (obj.orderedDrink) {                  result += ` and ${obj.orderedDrink}.`              }              return result;          }      },      getRemainingWork: function (statusArr) {          const remainingArr = statusArr.filter(s => s.status != 'ready');          if (remainingArr.length > 0) {              let pizzaNames = remainingArr.map(p => p.pizzaName).join(', ')              let pizzasLeft = `The following pizzas are still preparing: ${pizzaNames}.`              return pizzasLeft;          } else {              return 'All orders are complete!'          }      },      orderType: function (totalSum, typeOfOrder) {          if (typeOfOrder === 'Carry Out') {              totalSum -= totalSum \* 0.1;              return totalSum;          } else if (typeOfOrder === 'Delivery') {              return totalSum;          }      }  } |

**Submission**

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

const pizzUni = require('./unit');

const { assert } = require('chai');

describe("Tests …", function () {

    describe("TODO …", function () {

        it("Make an order", function () {

            let pizza = { orderedPizza: 'pizza', orderedDrink: 'drink' };

            let pizza1 = { orderedDrink: 'drink' };

            let pizza2 = { orderedPizza: 'pizza' };

            let pizza3 = {};

            assert.throw(() => pizzUni.makeAnOrder(pizza1), 'You must order at least 1 Pizza to finish the order.');

            assert.throw(() => pizzUni.makeAnOrder(pizza3), 'You must order at least 1 Pizza to finish the order.');

            assert.equal(pizzUni.makeAnOrder(pizza2), `You just ordered ${pizza2.orderedPizza}`);

            assert.equal(pizzUni.makeAnOrder(pizza), `You just ordered ${pizza.orderedPizza} and ${pizza.orderedDrink}.`);

        });

        it("getRemainingWork", function () {

            let statusMixed = [{ pizzaName: 'pizza', status: 'ready' },

            { pizzaName: 'pizza2', status: 'ready' },

            { pizzaName: 'pizza3', status: 'preparing' },

            { pizzaName: 'pizza4', status: 'preaparing' }]

            let statusReady = [{ pizzaName: 'pizza', status: 'ready' },

            { pizzaName: 'pizza2', status: 'ready' }];

            let statusNotReady = [{ pizzaName: 'pizza3', status: 'preparing' },

            { pizzaName: 'pizza4', status: 'preparing' }];

            assert.equal(pizzUni.getRemainingWork(statusReady), 'All orders are complete!');

            assert.equal(pizzUni.getRemainingWork(statusMixed), 'The following pizzas are still preparing: pizza3, pizza4.');

            assert.equal(pizzUni.getRemainingWork(statusNotReady), 'The following pizzas are still preparing: pizza3, pizza4.');

        });

        it("orderType", function () {

            let delivery = 'Delivery';

            let carryOut = 'Carry Out';

            let totalSum = 100;

            assert.equal(pizzUni.orderType(totalSum, delivery), 100);

            assert.equal(pizzUni.orderType(totalSum, carryOut), 90);

        });

    });

});