**03. Food Delivery**

**Your Task**

Using **Mocha** and **Chai** write **JS Unit Tests** to test a variable named **foodDelivery**, 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 should have the following functionality:

* **getCategory(category) -** A function that accepts one parameter: **string**.
* If the **category** is "**Vegan**" return the string:

**"Dishes that contain no animal products."**

* If the **category** is "**Vegetarian**" return the string:

**"Dishes that contain no meat or fish."**

* If the **category** is "**Gluten-Free**" return the string:

**"Dishes that contain no gluten."**

* If the **category** is "**All**" return the string:

**"All available dishes."**

* If the value of the string type is **different** from **"Vegan", "Vegetarian", "Gluten-Free", "All"**, **throw an error**:

**"Invalid Category!"**

* **addMenuItem(menuItem, maxPrice) -** A function that accepts an **array** of objects(**{name: Item name, price: item price}**) and **number**.
* You must **add** an element (**menuItem**) if the price is **less** or equal to **maxPrice** from the array to **availableItems** array.
* Finally, **return** the array **length** in the following string:
* **"There are {availableItems.length} available menu items matching your criteria!"**
* There is a need for validation for the input, as an **array** and

**number** may not always be valid. In case of submitted invalid parameters, **throw an error**

**"Invalid Information!"**

* If passed **menuItem** or **maxPrice** parameters are not an **array** and **number**.
* If the **menuItem** array has fewer than 1 item, and if **maxPrice** is **less** than 5.
* **calculateOrderCost(shipping, addons, discount) -** A function that accepts three parameters: **array**, **array**, and **boolean**.
  + Calculate the total **price** you are going to **pay** depending on the chosen **shipping** options and **addons**.
* The result must be formatted to the second digit after the decimal point.
* The available options for **shipping** are:
* **standard**, which costs $3
* **express**, which costs $5
* The available options for **addons** are:
* **sauce**, which costs $1
* **beverage**, which costs $3.5
* If the **discount** is **true**, a 15% discount should be applied. Then **return** the following message:

**"You spend ${totalPrice} for shipping and addons with a 15% discount!"**

* Else, **return** the following message:
* **"You spend ${totalPrice} for shipping and addons!"**
* You need to **validate** the input. If the **shipping**, **addons**, and **discount** are not an **array**, **array**, and **boolean**, **throw** an error:

**"Invalid Information!"**

* **Note**: Тhe **totalPrice** must be rounded to the **second** decimal

**JS Code**

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

|  |
| --- |
| foodDelivery.js |
| const foodDelivery = {    getCategory(category) {      if (category === "Vegan") {        return "Dishes that contain no animal products.";      } else if (category === "Vegetarian") {        return "Dishes that contain no meat or fish.";      } else if (category === "Gluten-Free") {        return "Dishes that contain no gluten.";      } else if (category === "All") {        return "All available dishes.";      } else {        throw new Error("Invalid Category!");      }    },    addMenuItem(menuItem, maxPrice) {      if (        !Array.isArray(menuItem) ||        typeof maxPrice !== "number" ||        menuItem.length < 1 ||        maxPrice < 5      ) {        throw new Error("Invalid Information!");      }      let availableItems = [];      menuItem.forEach((item) => {        if (item.price <= maxPrice) {          availableItems.push(item);        }      });      return `There are ${availableItems.length} available menu items matching your criteria!`;    },    calculateOrderCost(shipping, addons, discount) {      if (        !Array.isArray(shipping) ||        !Array.isArray(addons) ||        typeof discount !== "boolean"      ) {        throw new Error("Invalid Information!");      }      let totalPrice = 0;      shipping.forEach((item) => {        if (item === "standard") {          totalPrice += 3;        } else if (item === "express") {          totalPrice += 5;        }      });      addons.forEach((item) => {        if (item === "sauce") {          totalPrice += 1;        } else if (item === "beverage") {          totalPrice += 3.5;        }      });      if (discount) {        totalPrice = totalPrice \* 0.85;        return `You spend $${totalPrice.toFixed(          2        )} for shipping and addons with a 15% discount!`;      } else {        return `You spend $${totalPrice.toFixed(2)} for shipping and addons!`;      }    },  }; |

**Submission**

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