# JavaScript OOP – Practical Exam

7 June 2014

## Task Description

Implement the functionality for a Tech Store. A store can have many items to sell. You are given a part of the application and you should **only implement the types Item and Store**.

## Items Description

Items represent the items in stock in a Store. Every item has **type, name and price:**

* **Type is a regular string** that can have any of the following values: 'accessory', 'smart-phone', 'notebook', 'pc' or 'tablet'. **These are the only possible values**. ***Any other value is invalid***
* **Name is a regular string** ***between 6 and 30-characters***-long
* **Price is a decimal floating-point number**

## Stores Description

Stores represent the store objects in the application. Stores keep a list of the items they have in stock. **Every store has a name that is a regular string** with length ***between 6 and 30 characters***.

Stores have the following behavior:

* **storeInstance.addItem(item)** – adds an item to the stock of the store. A store can **keep in stock only items of type Item**
* **storeInstance.getAll() –** returns a collection of all items, **sorted alphabetically**
* **storeInstance.getSmartPhones()** – returns a collection of only the items in stock that have **type 'smart-phone'**, **sorted alphabetically** by the name of the items
* **storeInstance.getMobiles()** – returns a collection of only the items in stock that have **type either 'smart-phone' or 'tablet'**, **sorted alphabetically** by the name of the items
* **storeInstance.getComputers()** – returns a collection of only the items in stock that have **type either 'pc' or 'notebook'**, **sorted alphabetically** by the name of the items
* **storeInstance.filterItemsByType(filterType)** – returns a collection of only the items in stock that have **type equal to the given filterType** (item.type === filterType), **sorted alphabetically** by the name of the items
* **storeInstance. filterItemsByPrice (options)** – returns a collection of only the items that **have a price from the price range** in the options parameter, **sorted ascending** by the price of the items. **The options object is optional and have optional properties min and max**.
  + If min is missing, it should be considered as 0
  + If max is missing, it should be considered +\infty
* **storeInstance.countItemsByType()** – returns an associative array that have **as keys the types**, that are of items in stock in the store, and **values that are equal to the number of items** with this type
* **storeInstance**.**filterItemsByName**(partOfName) – returns a collection of only the items in stock that have a **name containing partOfName**, **sorted alphabetically** by the name of the items. The search should be performed case insensitive

**Your task is to implement both the Item and Store modules, using RequireJS and classical JavaScript OOP. Use the best practices for Classical OOP in JavaScript.** After your implementation, **the code in the app.js file should work.**

## Constraints

* You are allowed only to change **the contents of the files "item.js" and "store.js**" in the   
  "tech-store-models" folder

## Files to Submit

When ready, send **all files in the task-files folder** as a single zip file.