# Introduction to Node.js Exercises

## Storage

Prepare the "**package.json**" file and the IDE configurations. Create a module which is named "**storage**". The purpose of the module is to **store** key-value pairs where the key is always a string. Export from the module the following functions: "**put**", "**get**", "**update**", "**delete**", "**clear**", "**save**" and "**load**".

* put(key, value) – The "**put**" function should have two parameters – one for the **key** and one for the **value**. If the key is not a string, you should throw an error. If the key already **exists** in the storage, you should throw an error. Otherwise you should save the key-value pair in memory
* get(key) – The "**get**" function should have one parameter – for the key. If the key is not a string, you should throw an error. If the key does not exist in the storage, you should throw an error. Otherwise the function should **return** the **value** corresponding to the provided key
* getAll() – The "**getAll**" function should return all key-value pairs from the storage, if the storage is empty it should print appropriate message
* update(key, newValue) – The "**update**" function should have two parameters – one for the key and one for the value. If the key is not a string, you should throw an **error**. If the key does not exist in the storage, you should throw an **error**. Otherwise you should **update** the **key-value pair** in memory
* delete(key) – The "**delete**" function should have one parameter – for the key. If the key is not a string, you should throw an error. If the key does not exist in the storage, you should throw an error. Otherwise you should **delete** the **key-value pair** from the memory storage
* clear() – The "**clear**" function should **delete** **all** saved key-value pairs **in** the **storage**.
* save() – The "**save**" function should **save** all key-value pairs **on a file** named "**storage.json**". Use JSON format for saving the data. Every time the "save" function is called the file should be overridden, starting from a blank state. Use **synchronous** file access
* load() – The "**load**" function should read a file named "**storage.json**", parse the data, and load all the key-value pairs in memory. If the file does not exist yet, do nothing. Use **synchronous** file access

### Example

Create an "**index.js**" file and import the storage module you just created. Use it in code to see whether or not the storage module is working correctly, you can also use the ones provided below.

|  |  |  |
| --- | --- | --- |
| Sample code usage |  | Corresponding output |
| storage.load()  storage.put('first','firstValue')  storage.put('second','secondValue')  storage.put('third','thirdValue')  storage.put('fouth','fourthValue')  console.log(storage.get('first'))  console.log(storage.getAll())  storage.delete('second')  storage.update('first','updatedFirst')  storage.save()  storage.clear()  console.log(storage.getAll())  storage.load()  console.log(storage.getAll()) | firstValue  { first: 'firstValue',  second: 'secondValue',  third: 'thirdValue',  fouth: 'fourthValue' }  There are no items in the storage  { first: 'updatedFirst',  third: 'thirdValue',  fouth: 'fourthValue' } |

|  |
| --- |
| **Sample code usage** |
| storage.put('first','firstValue')  storage.put('second','secondValue')  storage.delete('second')  storage.delete('second') |
| storage.put(2,'someValue') |
| storage.put('cat','dog')  storage.put('cat','anotherDog') |

|  |
| --- |
| Corresponding output |
| All examples should throw appropriate Error |

**\*** You **must** **have** variable named **"storage"** that is **connected** to the **storage module,** also you **must delete** the **"storage.json"** file, before testing.

## Asynchronous Storage

Rewrite the previous task to work **asynchronously**. Add a **callback** parameter to all functions and make them **execute the callback with the data**, instead of returning it directly.