# Dictionary

Dictionaries are key-value pairs used for easy lookups. They are also called associative arrays as each entry into the dictionary is a key with an associated value. In this tutorial, we look at how to implement dictionaries in Javascript, how to handle duplicate keys and how to implement lookups and deletes.

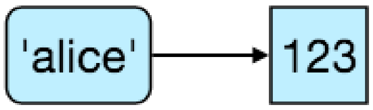
## Introduction

Dictionary is a data structure that consists of key-value pairs. It's a data structure where a key is associated with a value. Hence, they are also called ***associative arrays***. Dictionaries provides lookups on its keys.

Let's go through a quick visualization

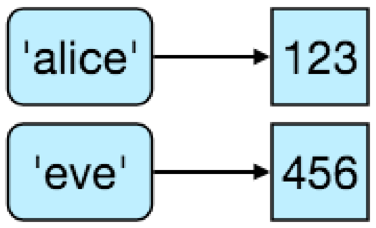
**Let's save phone's of Alice, Eve and Bob in a dictionaryof**

**1** of 4



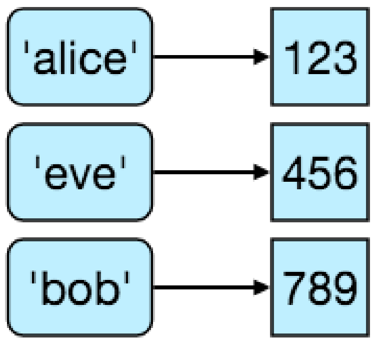
**Alice's phone is 123**

**2** of 4



Eve's phone is 456

**3** of 4



Bob's phone is 789

**4** of 4

## Duplicates in Dictionary

We are not going to handle duplicates in our dictionary. Most implementations of associative arrays don't support duplicates as it is mostly thought of as a 1:1 relationship between a key and a value instead of a 1:N relationship. However, if you are in a situation where you need to associate duplicate values to a key, it's not that difficult to implement either.

## Key Types

For simplicity, we are going only to accept strings or integers as keys. Why not other types of keys? We can do it, but we have to then define a standard way other types of keys? We can do it, but we have to then define a standard way

to compare and equate objects. For now, we'll keep it simple as Javascript already provides a way to compare integers and strings.

We are also going to consider an integer and its string representation as the same key e.g. 45 and "45" will refer to the same key.

## Why don't we use Javascript Object?

You are right. Javascript object is already an associative array. Why don't we use it? We will use it internally but with our implementation, we can provide cleaner methods around our Dictionary e.g. we can provide a length method. We can also provide methods to enumerate keys and values.

Let's start implementing the dictionary.

## Implementing Dictionary in Javascript

We'll use Javascript's object for our key/value pair storage.

```
function Dictionary() {
  this._data = {};
  this._length = 0;
}
```

## Adding a element to the dictionary

First we'll ensure  that the key is a string or an integer. In addition, if someone tries to add a key that already exists, we will throw an exception (this is again a matter of preference and the other option is to overwrite the value). We use ***Object.hasOwnProperty*** method to check for duplicates.

```
Dictionary.prototype.add = function(key, value) {
  var keyType = typeof(key);

  if (keyType !== 'string'  && keyType !== 'number') {
    throw 'Key type can only be string or a number';
  }

  if (this._data.hasOwnProperty(key)) {
    throw 'Duplicate keys are not supported';
  }
  this._data[key] = value;

  this._length++;
}
```

## Looking up a value in the dictionary

Next let's look how to implement the lookup method. It's pretty straightforward.

```
Dictionary.prototype.find = function(key) {
  if (key === null) {
    return undefined;
  }

  var keyType = typeof(key);
  if (keyType !== 'string'  && keyType !== 'number') {
    return undefined;
  }

  if (this._data.hasOwnProperty(key)) {
    return this._data[key];
  }

  return undefined;
}
```

## Removing an element from the Dictionary

We'll use ***delete*** function to remove both the key and its associated value from our data object. Here's the code for that.

```
Dictionary.prototype.remove = function(key) {
  if (this._data.hasOwnProperty(key)) {
        delete this._data[key];
        this._length--;
  }
}
```

## Dictionary in action

***> Run the following code to see dictionary in action***

JavaScript

HTML

CSS (SCSS)

```
var dict = new Dictionary();
dict.add("alice", 123);
dict.add("eve", 456);
dict.add("bob", 789);

console.log('dict.find(\'alice\') = '
            + dict.find('alice'));
console.log('dict.find(\'foo\') = '
            + dict.find('foo'));
```

Console

Clear

```
dict.find('alice') = 123
dict.find('foo') = undefined
```

## Exercise 1

> ***Implement the size method to fix the code below***

JavaScript

HTML

CSS (SCSS)

```
Dictionary.prototype.size = function() {
  // Implement this function
}

// Don't edit.
// This runs tests
runEvaluation();
```

Console

Clear

```
*** There is some bug lurking there. See failed test cases ***
```

```
Here are the tests that ran:

Test case FAILED for dict.size(). Result: undefined. Expected: 0
Test case FAILED for dict.size(). Result: undefined. Expected: 1
Test case FAILED for dict.size(). Result: undefined. Expected: 1
```

## Summary

Dictionary is a collection of key/value pairs.

It's also called an associative array as every key is associated with a value.

Some dictionary implementations support multiple values associated with a single key.

Some dictionary implementations allow overwriting the value associated with the key. Others don't allow this in which case you would have to first delete the key and insert it again with the new value.