

CSE 503

Introduction to Computer Science for Non-Majors

Dr. Eric Mikida

epmikida@buffalo.edu

208 Capen Hall

Day 14

Associative Collections in JavaScript

Announcements

- Slight office hours change for Wednesdays
- Monday – Come prepared with questions/examples!

Recap

Dictionaries are a type of associative collection in Python

They are a collection of **key:value pairs**:

```
dict = {"name": "Eric", "job": "Lecturer"}
```

Values can be *accessed*, *added*, and *updated* via a key using square brackets []:

```
dict["age"] = 32
```

We can *remove* keys using **del** or **pop**:

```
del d["age"]
```

We can *test* if a key exists in a dictionary using **in** or **not in**:

```
"name" in d
```

DNA Frequency Example

Write a function called `dnaFrequency` that takes a single DNA string, and returns a *dictionary* containing the frequency of each base.

For example:

```
dnaFrequency("ACAGCCTAAG") must return  
{"A": 4, "C": 3, "G": 2, "T": 1}
```

How does this compare to the list version?

Associative Collections in JavaScript

- JavaScript also has associative collections for storing key:value pairs
- They come in two varieties: **Objects** and **Maps**
 - **Objects**: Simpler, but more restrictive. Direct JSON support.
 - **Maps**: More complex, richer operations. No JSON support.
- For now, our focus will be on **Objects**

Object: Operations

Creation:

```
let x = {};
```

```
let y = {'a':1, 'b':2, 'c':3, 'd':4};
```

Object: Operations

Creation:

```
let x = {};  
let y = {'a':1, 'b':2, 'c':3, 'd':4};
```

Update/Add/Access:

```
y['c'] = 12; // Can use an expression...  
y.b = 7;    // ...or a literal as the key  
y['z'] = 3;  
console.log(y['c'])  
console.log(y.c)
```

Object: Operations

Creation:

```
let x = {};  
let y = {'a':1, 'b':2, 'c':3, 'd':4};
```

Update/Add/Access:

```
y['c'] = 12; // Can use an expression...  
y.b = 7;    // ...or a literal as the key  
y['z'] = 3;  
console.log(y['c'])  
console.log(y.c)
```

Updating existing values



Object: Operations

Creation:

```
let x = {};  
let y = {'a':1, 'b':2, 'c':3, 'd':4};
```

Update/Add/Access:

```
y['c'] = 12; // Can use an expression...  
y.b = 7;    // ...or a literal as the key
```

```
y['z'] = 3;
```

```
console.log(y['c'])
```

```
console.log(y.c)
```

← Adding a new key:value pair

Object: Operations

Creation:


```
let x = {};  
let y = {'a':1, 'b':2, 'c':3, 'd':4};
```

Update/Add/Access:

```
y['c'] = 12; // Can use an expression...  
y.b = 7;    // ...or a literal as the key  
y['z'] = 3;
```

```
console.log(y['c'])  
console.log(y.c)
```

Accessing (and printing) the value of existing key:value pairs



Object: Operations

Removal:

```
delete y['c']
```

```
delete y.c
```

Object: Operations

Removal:

```
delete y['c']  
delete y.c
```

Membership Test:

```
'c' in x  
!('c' in x)
```

Object: Components

Direct Access to All Keys, Values, and Pairs:

```
Object.keys(y);
```

```
Object.values(y);
```

```
Object.entries(y);
```

Exercise #1

Write a function, `valueCount`, that given a dictionary and a value, counts the number of times that the value shows up in the dictionary.

Examples:

```
valueCount({}, 32) # Should return 0
```

```
valueCount({"Eric":32,"Alicia":30,"Cory":30},30) # Should return 2
```

Exercise #2

Write a function `getKeysFor` that takes a dictionary and a value, and returns a list of all the keys in the dictionary that have that value.

Examples:

```
getKeysFor({"Eric":32, "Alicia":30, "Cory":30}, 30)
```

```
# Should return ["Alicia", "Cory"]
```

```
getKeysFor({"Eric":32, "Alicia":30, "Cory":30}, 29)
```

```
# Should return []
```