Associative Arrays

A Key-Value Pair Structure

Table of Contents

Associative Arrays

Storing Key-Value Pairs

What is an Associative Array ?

- Arrays indexed by **string keys**
- Hold a set of pairs **[key => value]**
  - The key can either be an **integer** or a **string**
  - The **value** can be of **any** type

Key	Value
John Smith	+1-555-8976
Lisa Smith	+1-555-1234
Sam Doe	+1-555-5030

Declaration

- An associative array in JavaScript is just an **object**
- We can declare it **dynamically**

```
let assocArr = {
 'one': 1,
 'two': 2,
 'three': 3
};
```

Attributes

- The syntax for **accessing** the value of a key is

```
assocArr['key'] // person['age']
```

or

```
assocArr[key] // key = "age"; person[key]
```

- **Assigning** a value to a variable

```
let age = assocArr[key];
```

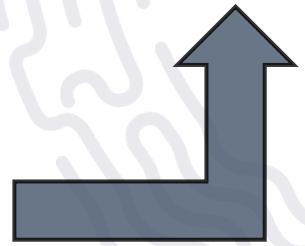
Using for – in

- We can use **for-in** loop to iterate through the keys

```
let assocArr = {};
assocArr['one'] = 1;
assocArr['two'] = 2;
assocArr['three'] = 3;

for(let key in assocArr) {
   console.log(key + " = " + assocArr[key]);
}
```

```
// one = 1
// two = 2
// three = 3
```



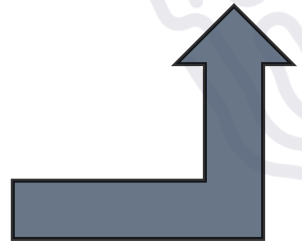
Using ForEach

- We can also use **forEach** loop to iterate through the **keys**

```
let assocArr = {};
assocArr['one'] = 1;
assocArr['two'] = 2;
assocArr['three'] = 3;

Object.keys(assocArr).forEach( i => {
  console.log(`${i} = ${assocArr[i]}`));
}
```

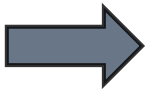
```
// one = 1
// two = 2
// three = 3
```



Problem: Phone Book

- Write a function that reads **names** and **numbers**
- Store them in an associative array and print them
- If same name occurs, safe the **latest** number

```
[`Tim 0834212554`,
`Peter 0877547887`,
`Bill 0896543112`,
`Tim 0876566344`]
```



```
Tim -> 0876566344
Peter -> 0877547887
Bill -> 0896543112
```

Solution: Phone Book

```
function solve(input) {
  let phonebook = {};
  for (let string of input) {
    let tokens = string.split(' ');
    let name = tokens[0];
    let number = tokens[1];
    phonebook[name] = number;
  }
  for (let key in phonebook) {
    console.log(`${key} -> ${phonebook[key]}`);
  }
}
```

Maps

Storing Key-Value Pairs

What is a Map?

- A **Map** object stores its elements in **insertion order**
- A for-of loop returns an array of **[key, value]** for each iteration
- Pure **JavaScript objects** are like **Maps** in that both let you:
  - Set **keys** to **values**
  - Delete keys
  - Detect whether something is stored in a key

Adding/Accessing Elements

- **`.set`(key, value) - adds a new key-value pair**
  - let map = new Map();
  - map.set(1, "one"); *// key - 1, value - one*
  - map.set(2, "two"); *// key - 2, value - two*

- **`.get`(key) - returns the value of the given key**

```
map.get(2); // two
map.get(1); // one
```

Contains / Delete

- **`.has(key)`** - checks if the map has the given key

```
map.has(2); // true
map.has(4); // false
```

- **`.delete(key)`** - removes a key-value pair

```
map.delete(1); // Removes 1 from the map
```

- **`.clear()`** - removes all key-value pairs

Iterators

- **`.entries()`** - returns Iterator - array of **[key, value]**
- **`.keys()`** - returns Iterator with all the **keys**
- **`.values()`** - returns Iterator with all the **values**

```
let entries = Array.from(map.entries());
// [ [2, 'two'], [3, 'three'] ]
let keys = Array.from(map.keys()); // [2, 3]
let values = Array.from(map.values()); // ['two', 'three']
```

These methods return an Iterator, transform it into an **Array**

Iterating a Map

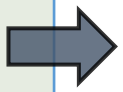
- To print a map simply use one of the **iterators** inside a **for-of**

```
let iterable = Array.from(phonebookMap.entries());
for(let kvp of iterable) {
  let name = kvp[0];
  let number = kvp[1];
  console.log(`${name} => ${number}`);
}
```

Problem: Storage

- Write a function that **stores products** and their **quantity**
- If the same product appears **more than once**, **add** the new quantity to the old one

```
tomatoes 10
coffee 5
olives 100
coffee 40
```



```
tomatoes -> 10
coffee -> 45
olives -> 100
```

Solution: Storage

```
let map = new Map();
for(let string of input) {
  let tokens = string.split(' ');
  let product = tokens[0];
  let quantity = Number(tokens[1]);
  if(!map.has(product)) {
    map.set(product, quantity);
  } else {
    let currQuantity = map.get(product);
    let newQuantity = currQuantity += quantity;
    map.set(product, newQuantity);
  }
}
// TODO: Print Map
```

Map Sorting

- To **sort** a Map, firstly transform it into an **array**
- Then use the **`sort()`** method

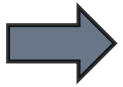
Sort **ascending** by value

```
let map = new Map();
map.set("one", 1);
map.set("eight", 8);
map.set("two", 2);
let sorted = Array.from(map.entries())
      .sort((a, b) => a[1] - b[1]);
for (let kvp of sorted) {
    console.log(`${kvp[0]} -> ${kvp[1]}`);
}
```

Problem: School Grades

- Write a function to **store students** with all their **grades**
- If a student appears **more than once** add the **new grades**
- At the end print the students sorted by **average grade**

```
[`Lilly 4 6 6 5`,
`Tim 5 6`,
`Tammy 2 4 3`,
`Tim 6 6`]
```



```
Tammy: 2, 4, 3
Lilly: 4, 6, 6, 5
Tim: 5, 6, 6, 6
```

Solution: School Grades

```
function solve(input) {
    let map = new Map();
    for(let string of input) {
        let tokens = string.split(' ');
        let name = tokens[0];
        let grades = tokens
              .splice(1, tokens.length).map(Number);
        // TODO: Fill the map
    }
    let sorted = Array.from(map).sort((a, b) => average(a, b));
    // TODO: Print each key and joined values
}
// TODO: Implement the average function
```


Set()
Sets
Storing Unique Elements

What is a Set?

- Store **unique values** of any type, whether **primitive** values or **object** references
- Set objects are **collections** of values
- Can **iterate** through the elements of a set in **insertion** order

```
let set = new Set([1, 2, 2, 4, 5]);
// Set(4) { 1, 2, 4, 5 }
console.log(set.has(1));
// Expected output: true
```

Summary

- We can use both **Arrays** and **Maps** to store **key-value** pairs
- **Maps** are a better way to do it because:
  - They are **iterable**
  - They have **size property**
  - They are better for **adding** and **deleting** many **key-value** pairs

Questions?

License

- This course (slides, examples, demos, exercises, homework, documents, videos and other assets) is **copyrighted content**
- Unauthorized copy, reproduction or use is illegal
- © Kingsland University – https://kingslanduniversity.com


KINGSLAND
UNIVERSITY
THANK YOU