JavaScript Map/Set vs Python Dict/Set Cheat Sheet

Quick Comparison Table

Feature	JavaScript Map	Python Dict	JavaScript Set	Python Set
Declaration	`new Map()`	`{}` or `dict()`	`new Set()`	`set()`
Key Types	Any type	Hashable types	Any type	Hashable types
Order	Insertion order	Insertion order (Python 3.7+)	Insertion order	Unordered
Size	`.size`	`len()`	`.size`	`len()`

JavaScript Map vs Python Dictionary

Creation

javascript

```
// JavaScript Map
const jsMap = new Map();
const jsMapWithValues = new Map([['a', 1], ['b', 2]]);
```

python

```
# Python Dictionary
py_dict = {}
py_dict = {'a': 1, 'b': 2}
py_dict = dict(a=1, b=2)
```

Basic Operations

javascript

```
// JavaScript Map
jsMap.set('key', 'value');        // Add/update
jsMap.get('key');                 // Retrieve
jsMap.has('key');                 // Check existence
jsMap.delete('key');              // Remove
jsMap.clear();                    // Remove all
jsMap.size;                       // Get size
```

python

```
# Python Dictionary
py_dict['key'] = 'value'          # Add/update
py_dict['key']                    # Retrieve
'key' in py_dict                  # Check existence
py_dict.get('key')                # Safe retrieve
del py_dict['key']                # Remove
py_dict.pop('key')                # Remove & return
py_dict.clear()                   # Remove all
len(py_dict)                      # Get size
```

Iteration

javascript

```
// JavaScript Map
for (let [key, value] of jsMap) { }
jsMap.forEach((value, key) => { });
for (let key of jsMap.keys()) { }
for (let value of jsMap.values()) { }
```

python

```
# Python Dictionary
for key in py_dict:               # Iterate keys
for key, value in py_dict.items(): # Iterate key-value pairs
for value in py_dict.values():    # Iterate values
```

JavaScript Set vs Python Set

Creation

javascript

```
// JavaScript Set
const jsSet = new Set();
const jsSetWithValues = new Set([1, 2, 3, 3, 4]); // {1, 2, 3, 4}
```

python

```
# Python Set
py_set = set()
py_set = {1, 2, 3, 3, 4}         # {1, 2, 3, 4}
```

Basic Operations

javascript

```
// JavaScript Set
jsSet.add(value);                 // Add element
jsSet.has(value);                 // Check existence
jsSet.delete(value);              // Remove element
jsSet.clear();                    // Remove all
jsSet.size;                       // Get size
```

python

```
# Python Set
py_set.add(value)                  # Add element
value in py_set                    # Check existence
py_set.remove(value)               # Remove (error if missing)
py_set.discard(value)              # Remove (no error)
py_set.clear()                     # Remove all
len(py_set)                        # Get size
```

Set Operations

javascript

```
// JavaScript Set Operations
// Union
new Set([...setA, ...setB]);

// Intersection
new Set([...setA].filter(x => setB.has(x)));

// Difference
new Set([...setA].filter(x => !setB.has(x)));
```

python

```
# Python Set Operations
setA | setB          # Union
setA & setB          # Intersection
setA - setB          # Difference
setA ^ setB          # Symmetric Difference
setA.union(setB)     # Union method
setA.intersection(setB) # Intersection method
setA.difference(setB) # Difference method
```

Iteration

javascript

```
// JavaScript Set
for (let item of jsSet) { }
jsSet.forEach(value => { });
```

python

```
# Python Set
for item in py_set:  # Iterate elements
```

Key Differences

1. Key Requirements

- **JavaScript**: Maps can use any type as keys (objects, functions, etc.)

- **Python**: Dictionary keys must be hashable (immutable types)

2. Accessing Non-existent Keys

javascript

```
// JavaScript Map
jsMap.get('nonexistent'); // Returns undefined
```

python

```
# Python Dictionary
py_dict['nonexistent']    # Raises KeyError
py_dict.get('nonexistent') # Returns None
py_dict.get('nonexistent', 'default') # Returns 'default'
```

3. Order Guarantees

- **JavaScript**: Maps and Sets maintain insertion order
- **Python**: Dictionaries maintain insertion order (Python 3.7+), Sets are unordered

4. Built-in Methods

- **Python** has more built-in set operations
- **JavaScript** requires manual implementation for some set operations

Performance Notes

- Both have average O(1) time complexity for lookups, insertions, and deletions
- JavaScript Maps are better for frequent additions/removals
- Python dictionaries are highly optimized for most use cases

Common Patterns

Converting to Array/List

javascript

```
// JavaScript
Array.from(jsMap.keys());
Array.from(jsMap.values());
[...jsSet]; // Set to Array
```

python

```
# Python
list(py_dict.keys())
list(py_dict.values())
list(py_set)          # Set to List
```

Object/Map Conversion

javascript

```
// JavaScript - Map from Object
const mapFromObj = new Map(Object.entries(obj));

// JavaScript - Object from Map
const objFromMap = Object.fromEntries(map);
```

python

```
# Python - No direct equivalent needed since dicts are primary
```