****

JavaScript Cheat Sheet

****

1. **Array:**

- Loop:

  for (let i = 0; i < array.length; i++) { /* code */ }

- Add: array.push(value)

- Remove: array.pop()

- Get: array[index]

- Set: array[index] = value

- Check Existence: array.includes(value)

2. **List:**

- Loop:

  for (let i = 0; i < array.length; i++) { /* code */ }

- Add: AppendItem(item) {

    this.items.push(item);

    this.length = this.items.length;

  }

- Remove: RemoveItem(element) {

    if (this.CheckItemExists(element)) {

      this.items = this.items.filter(item => item !== element);

      this.length = this.items.length;

    }

- Get: GetItem(index) {

    if (index >= this.length) {

      return 0;

    }

- Set: constructor () {
    this.length = 0;
    this.items = [];
  }
- Check Existence: CheckItemExists(item) {
    return this.items.includes(item);
  }

3. **Dictionary:**

```
AddItem(key, value) {
    this.dictionary[key] = value;
  }
  GetItem(key) {
    return this.dictionary[key];
  }
  RemoveItem(value) {
    for (const key in this.dictionary) {
      if (this.dictionary[key] === value) {
        delete this.dictionary[key];
        return true;
      }
    }
    return false;
  }
  ExistValue(value) {
    for(var key in this.dictionary) {
      if (this.dictionary[key] === value) {
        return true;
      }
```

4. **Sorted List** (Not Native in JavaScript):

```
// Add an item
AddItem(key, value) {
    if (!this.data[key]) {
        this.keys.push(key);
        this.keys.sort();
    }
    this.data[key] = value;
}
// Remove an item
RemoveByKey(key) {
    const index = this.keys.indexOf(key);
    if (index !== -1) {
        this.keys.splice(index, 1);
        delete this.data[key];
    }
}
// Get an item
GetByKey(key) {
    return this.data[key];
}
// Check Existence (Key)
containsKey(key) {
    return this.keys.includes(key);
}
```

5**. HashSet** (Not Native in JavaScript):

```
// Add an item
AddItem(item) {
    this.data.add(item);
}
```

```
    // Remove an item
    RemoveItem(item) {
        this.data.delete(item);
    }
    // Check Existence of an item
    ContainsItem(item) {
        return this.data.has(item);
    }
```

6. **SortedSet** (Not Native in JavaScript):

```
// Add
    add(item) {
        if (!this.data.has(item)) {
            this.data.add(item);
            this.sortedArray.push(item);
            // Maintain a sorted order
            this.sortedArray.sort();
        }
    }
    // Remove
    remove(item) {
        if (this.data.has(item)) {
            this.data.delete(item);
            const index = this.sortedArray.indexOf(item);
            if (index !== -1) {
                this.sortedArray.splice(index, 1);
            }
        }
    }
```

```
  // Get
  get(index) {
    return this.sortedArray[index];
  }
  // Check Existence
  contains(item) {
    return this.data.has(item);
  }
```

7. **Queue:**

```
// Add an item
  enqueue(item) {
    this.data.push(item);
  }

 // Remove an item
  dequeue() {
    return this.data.shift();
  }
  // Get an item(Peek)
  peek() {
    return this.data[0];
  }
```

8. **Stack:**

```
// Add an item(Push)
 push(item) {
  this.data.push(item);
 }
 // Remove an item(Pop)
 pop() {
  return this.data.pop();
 }
```

```
  // Get an item(Peek)
  peek() {
    return this.data[this.data.length - 1];
  }
```

// Loops (Break/Continue)

---

1. For Loop:

```
  for (let i = 0; i < length; i++) {
     // code
     if (condition) break; // to exit loop
     if (condition) continue; // to skip to next iteration
  }
```

2. While Loop:

```
  while (condition) {
     // code
     if (condition) break;
     if (condition) continue;
  }
```

3. Do-While Loop:

```
  do {
     // code
     if (condition) break;
     if (condition) continue;
  } while (condition);
```

****

C# Cheat Sheet

****

1. **Array:**

- Loop:

  for (int i = 0; i < array.Length; i++) { /* code */ }

- Add: Array.Resize(ref array, array.Length + 1); array[array.Length - 1] = value;

- Remove: Array.Resize(ref array, array.Length - 1);

- Get: array[index]

- Set: array[index] = value

- Check Existence: Array.IndexOf(array, value) != -1

2. **List:**

- Loop: foreach (var item in list) { /* code */ }

- Add: list.Add(value)

- Remove: list.Remove(value) or list.RemoveAt(index)

- Get: list[index]

- Set: list[index] = newValue

- Check Existence: list.Contains(value)

3. **Dictionary:**

- Loop: foreach (var key in dictionary.Keys) { /* code */ }

- Add: dictionary.Add(key, value)

- Remove: dictionary.Remove(key)

- Get: dictionary[key]

- Set: dictionary[key] = newValue

- Check Existence: dictionary.ContainsKey(key)

4. **SortedList:**

- Loop: foreach (var key in sortedList.Keys) { /* code */ }
- Add: sortedList.Add(key, value)
- Remove: sortedList.Remove(key)
- Get: sortedList[key]
- Set: sortedList[key] = newValue
- Check Existence: sortedList.ContainsKey(key)

5. **HashSet:**

```
  //Add item
for (int i = 0; i < 10; i++)
{
   String value = String.Format("{0}{1}", "String", r.Next(0, 100));
   hashset.Add(value);
}
hashset.Add("CustomString1");
hashset.Add("CustomString2");
//Remove item by value
String valueToRemove = "CustomString2";
hashset.Remove(valueToRemove);
//Get item by key
String valueToSearch = "CustomString1";
String outValue = "";
hashset.TryGetValue(valueToSearch, out outValue);
```

6. **SortedSet:**

```
  //Add item
for (int i = 0; i < 10; i++)
{
   String value = String.Format("{0}{1}", "String", r.Next(0, 100));
```

```
    sortedset.Add(value);
}
sortedset.Add("CustomString1");
sortedset.Add("CustomString2");
//Remove item by value
String valueToRemove = "CustomString2";
sortedset.Remove(valueToRemove);
//Get item by key
String valueToSearch = "CustomString1";
String outValue = "";
sortedset.TryGetValue(valueToSearch, out outValue);
```

7. **Queue:**
   - Queue<T> queue = new Queue<T>();
   - Enqueue: queue.Enqueue(value)
   - Dequeue: queue.Dequeue()

8. **Stack:**
   - Stack<T> stack = new Stack<T>();
   - Push: stack.Push(value)
   - Pop: stack.Pop()

9. **LinkedList:**
   - LinkedList<T> linkedList = new LinkedList<T>();
   - AddLast: linkedList.AddLast(value)
   - Remove: linkedList.Remove(value)
   - Find: linkedList.Find(value)

// Loops (Break/Continue)

---

1. For Loop:

```
for (int i = 0; i < length; i++) {
  // code
  if (condition) break; // to exit loop
  if (condition) continue; // to skip to next iteration
}
```

2. While Loop:

```
while (condition) {
  // code
  if (condition) break;
  if (condition) continue;
}
```

3. Do-While Loop:

```
do {
  // code
  if (condition) break;
  if (condition) continue;
} while (condition);
```