# Chapter 77: WeakSet

## Section 77.1: Creating a WeakSet object

The WeakSet object is used for storing weakly held objects in a collection. The difference from Set is that you can't store primitive values, like numbers or string. Also, references to the objects in the collection are held weakly, which means that if there is no other reference to an object stored in a WeakSet, it can be garbage collected.

The WeakSet constructor has an optional parameter, which can be any iterable object (for example an array). All of its elements will be added to the created WeakSet.

```
const obj1 = {},
      obj2 = {};

const weakset = new WeakSet([obj1, obj2]);
```

## Section 77.2: Adding a value

To add a value to a WeakSet, use the `.add()` method. This method is chainable.

```
const obj1 = {},
      obj2 = {};

const weakset = new WeakSet();
weakset.add(obj1).add(obj2);
```

## Section 77.3: Checking if a value exists

To check if a value exits in a WeakSet, use the `.has()` method.

```
const obj1 = {},
      obj2 = {};

const weakset = new WeakSet([obj1]);
console.log(weakset.has(obj1)); // true
console.log(weakset.has(obj2)); // false
```

## Section 77.4: Removing a value

To remove a value from a WeakSet, use the `.delete()` method. This method returns `true` if the value existed and has been removed, otherwise `false`.

```
const obj1 = {},
      obj2 = {};

const weakset = new WeakSet([obj1]);
console.log(weakset.delete(obj1)); // true
console.log(weakset.delete(obj2)); // false
```