ES6 Sets

set functions and their usage

A `Set` data structure in ES6 is an *ordered* list of unique elements. Here are some set functions and their usage:

Keyword	Type	Usage
`Set`	constructor	creates a set
`add`	method	adds elements to the set
`size`	property	check size of the set
`has`	method	check if an element is a member of the set
`delete`	method	remove a value from a set

Consider the following code:

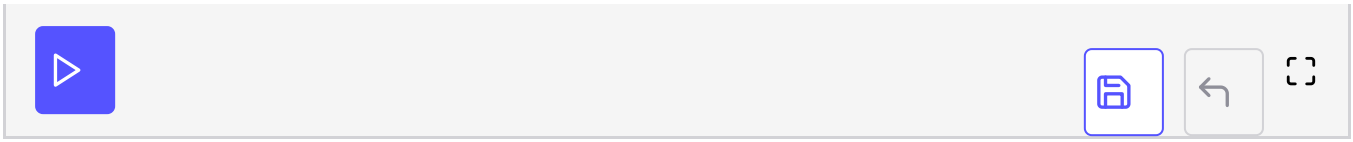
```
let colors = new Set();

colors.add( 'red' );
colors.add( 'green' );
colors.add( 'red' );   // duplicate elements are added only once
console.log( colors );
//> Set {"red", "green"}

console.log( 'Size: ' + colors.size );
//> 2

console.log( 'has green: ' + colors.has( 'green' ) + '\nhas blue: ' + colors.has( 'blue' ) );
```

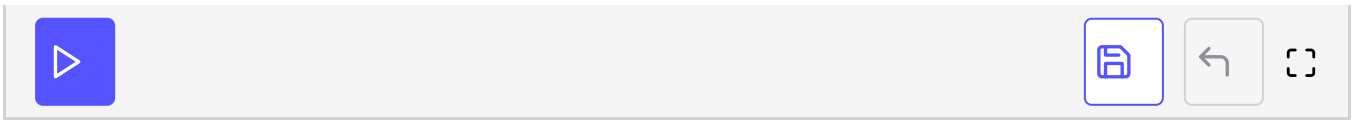
```
//> true false
```



You can remove a value from a set by calling its `delete` method. The return value of the removal is a boolean, indicating whether the removed element was initially a member of the set or not.

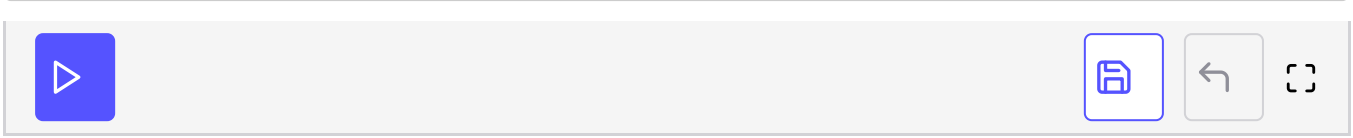
```
console.log('Before deleting: ')
console.log(colors);
colors.delete( 'green' )
//> true
colors.delete( 'green' )
//> false

console.log('\nAfter deleting: ')
console.log(colors);
```



The `Set` constructor accepts an optional array argument with initial values. It eliminates all duplicates.

```
let moreColors = new Set( ['red', 'blue', 'red', 'orange'] );
console.log( moreColors );
```



We will revisit the set construction when learning about iterators and generators.

Now, let's move on to set iteration.