Arrays

## Associative Arrays

- An associative array is effectively a map of key value pairs, i.e. an array where the index is a value of your choosing
- Associative arrays are NOT supported in JS (but you could be forgiven for thinking otherwise)
  e.g. `arr['name'] = 'Dave';`
- Arrays are objects; the [] syntax is an alternative means of accessing object properties, in fact, it's superior insofar as it permits expressions
- Custom array properties have no bearing on length, traversal or methods like push and pop

# Traversal

- The `for in` loop is designed for traversing the properties of an object; it can be used for arrays but will yield indexes (and other properties)

- The `for of` loop is designed for iterable objects (objects that implement the Iterator protocol), e.g. arrays, Sets, and Maps

- `forEach` is an iterable object method; it is passed a function which is called once for each element and that accepts the element as an arg.

## Deleting Elements

- The `delete` operator may be used to delete array elements, e.g. `delete arr[3];`
- Deleting an element (making it empty) in this way does not alter the array length
- Accessing the deleted element yields `undefined` except when using `forEach` which ignores empty elements
- NB: empty is not the same as undefined; forEach will not ignore undefined elements

## Length

- An array's `length` property is writable so may be used to truncate or extend the array, e.g.:

```
var arr = [1, 2, 3];
arr.length = 5;
// arr = [1, 2, 3, empty x 2]
```

# Summary

- JS does not support associative arrays; the [] syntax may be used to access object props
- The `for of` loop is designed for iterable objects
- The `forEach` iterable method accepts a function that will called once for each element; it ignores empty elements
- The `delete` keyword may be used to delete array elements without altering the length of the array
- An array's `length` property is writable