```javascript
'use strict';
prompt("Enter a number:" );
// Lecture 39: Introduction to Arrays
const friend1 = 'Michael';
const friend2 = 'Steven';
const friend3 = 'Peter';

// Initialising an array (literal syntax)
const friends = ['Michael', 'Steven', 'Peter'];
// Note that the variable name does not require
// the [] notation.
console.log(friends);
// Another way of initialising an array
// Remember to use the 'new' keyword
const y = new Array(1991, 1984, 2008, 2020);
// An array can hold any number of values that we want, and also values of any type that we like.
// Arrays in JS are zero-based.
console.log(friends[0]); // ==> Michael
console.log(friends[2]); // ==> Peter
// Obtain the number of elements in the array
// Access the length as a PROPERTY (OOP)
console.log(friends.length);
// Obtain the last element in the array
// The length of array is NOT 0-based.
console.log(friends[friends.length - 1]);
// Modifying an element
```

```javascript
friends[2] = 'Jay';
console.log(friends); // ==> ['Michael', 'Steven', 'Jay']
// NOTE: the array is declared with CONST, and it turns out that only primitive values are immutable. An array is NOT a primitive value, and so we can ALWAYS change it and mutate it. It works this way because of the way that JS stores things in memory. More on Memory Allocation and management in the future.

// BUT what we cannot do is
 friends = ['Bob', 'Alice']; // ERROR CODE - reassignment to constant variable.

// Arrays can actually hold values with DIFFERENT PRIMITIVE TYPES
// You can even put an array inside an array.
const firstName = 'Jonas';
const jonas = [firstName, 'Schmedtmann', 2037 - 1991, 'teacher', friends];
console.log(jonas); // prints the array
console.log(jonas.length); // ==> 5
console.log(jonas[jonas.length - 1][0])// ==> Michael

// Exercise
const calcAge = function (birthYeah) {
```

```
const calcAge = function (birthYeah) {
  return 2037 - birthYeah; // operation here expects a single value
}
const years = [1990, 1967, 2002, 2010, 2018];
// note that in this case the function calcAge is expecting an integer, not an array
console.log(years); // ==>NaN
// So you have to do this instead.
const age1 = calcAge(years[0]);
const age2 = calcAge(years[1]);
const age3 = calcAge(years[years.length - 1]);
console.log(age1, age2, age3);

const ages = [calcAge(years[0]), calcAge(years[1]), calcAge(years[years.length - 1])];
console.log(ages);

// Hmm looks like this is inefficient: KIV - loops
//////////////////////////////////////////////////
// Lecture 40: Basic Array Operations (Methods)
const friends = ['Michael', 'Steven', 'Peter'];
// Arrays in JS are dynamic, you are able to add elements to arrays.
```

```javascript
// 1. Insertion

// A) Append to end of array
const newLength = friends.push('Jay'); // note that even if its a const, you can change its length attribute!
console.log(friends); // ==> [ ..., 'Peter', 'Jay']
// This push function returns a value, and it is the length of the new array.
console.log(newLength); // ==> 4
// B) Add element to beginning of the array
const length = friends.unshift('John');
console.log(friends); // ==> ['John', 'Michael', ...]
// The unshift methods also returns a value - that is the length of the new array.
console.log(length); // ==> 5

// 2. Remove elements
// A) Remove from end of array
friends.pop(); // Last
const popped = friends.pop();
// friends.pop() returns the removed element.
console.log(popped); // ==> Peter
console.log(friends); // ==> [John, ... , Steven]

// B) Remove the first element in the
```

```javascript
array
friends.shift();
console.log(friends);

// 3. Find the index of an element (O(n) time?)
console.log(friends.indexOf('Steven')); // ==> 1
console.log(friends.indexOf('Bob')); // ==> -1 (flag for an element not in a array)

friends.push(23);
// 4. If the array contains an element
// ES6 // This will return true or false
console.log(friends.includes('Steven')); // ==> true
console.log(friends.includes('Bob')); // ==> false
console.log(friends.includes(23)); // ==> true
// This is actually using strict equality - (no type coercion)
console.log(friends.includes(`23`)); // ==> false

// Using .includes() to write Conditions
if (friends.includes('Steven')) {
  console.log('You have a friend called Steven');
}
```