```
'use strict';
// Lecture 42: Introduction to Objects
// Introduction to Objects
// Array recap
const jonasArray = [
  'Jonas', // Sometimes people write a
    data structure in multiple lines.
  'Schmedtmann', // This is how pretty
    JSON is formatted too!
  2037 - 1991,
  'teacher',
  ['Michael', 'Peter', 'Steven'] // Array
    within array!
];
// In an array, there is no way of giving
  each of these ATTRIBUTES a name, but we
  can only, exclusively access an element
  by its index (for the case of an array)
// So, how do we get around this? Create
  a data structure - called an OBJECT.

// Definition of an object (Object
  Literal Syntax)
const jonas = {
    // key : value
  firstName: 'Jonas',
  lastName: 'Schmedtmann',
  age: 2037 - 1991,
  job: 'teacher',
  friends: ['Michael', 'Peter', 'Steven']
}; // this is the third time we are
  seeing curly braces, the first time it
```

```
    was for an if statement or switch
    statement, second time was for a
    function, and then this. Ultimately,
    curly braces are meant to represent
    scope.
// We use curly braces to define the
    attributes (or properties) of a
    particular object.
// So objects in JS are represented as a
    bunch of key-value pairs. Each key
    represents a PROPERTY - so this object
    has 5 properties. This is the most
    fundamental concept of creating objects.

// We use objects different variables
    that belong together, such as the
    properties of Jonas. The order of these
    objects don't matter, unlike that of
    the array.

////////////////////////////////////////
// Lecture 43: Dot vs. Bracket Notation
const jonas = {
  firstName: 'Jonas',
  lastName: 'Schmedtmann',
  age: 2037 - 1991,
  job: 'teacher',
  friends: ['Michael', 'Peter', 'Steven']
};
console.log(jonas);

// Accessing a property of jonas
```

```
// Accessing a property of jonas
// value = object.property
console.log(jonas.lastName); // Dot notation
 // An operation is an expression, so we can put this inside brackets.
console.log(jonas['lastName']);// Bracket notation
 // It will put any expression that we'd like.
const nameKey = 'Name';
console.log(jonas['first' + nameKey]); // ==> Jonas
console.log(jonas['last' + nameKey]);
//==> Schmedtmann
// note the concatenation happening before the accessing of the property.
// The concatenation will only work with the bracket notation. In dot notation, we will only be able to use the final property name, while in square bracket[] notation, we are allowed to use both the final property name and the computed property name.

// The following will result in wrong code:
// console.log(jonas.'last' + nameKey); / / ERROR CODE!!

// This will create a popup window with an input field
```

```
const interestedIn = prompt('What do you want to know about Jonas? Choose between firstName, lastName, age, job, and friends');

if (jonas[interestedIn]) {
  console.log(jonas[interestedIn]); // note that we HAVE to use the square brackets here.
  // also note that whatever the input it is A STRING, so we will not have to do a type conversion.
} else {
  console.log('Wrong request! Choose between firstName, lastName, age, job, and friends');
}

// Adding a Property
jonas.location = 'Portugal'; // via dot notation
jonas['twitter'] = '@jonasschmedtman'; // via square bracket notation
console.log(jonas);

// Challenge
// "Jonas has 3 friends, and his best friend is called Michael"
console.log(`${jonas.firstName} has ${jonas.friends.length} friends, and his best friend is called ${jonas.friends[0]}`);
```

```
// Note: Why does this work? This is because the member accessing in JS is of rather high priority, and it also acts from left to right, so (jonas.friends) will be evaluated first, and the resultant array.length will be accessed next, before the curly braces are created.

////////////////////////////////////////
// Lecture 44: Object Methods
// They can hold objects inside objects.
// Remember that functions are another type of value? So if functions are just another kind of value, we can create a key-value pair where the value is a function. So we can add functions to objects, and this gives us METHODS.

const jonas = {
  firstName: 'Jonas',
  lastName: 'Schmedtmann',
  birthYeah: 1991,
  job: 'teacher',
  friends: ['Michael', 'Peter', 'Steven'],
  hasDriversLicense: true,

  // Adding a method
  // Function Expression!! You HAVE to do this, you are not allowed to do the standard declaration, because you are not allowed to declare functions
```

```
    within objects However, you are
    allowed to declare methods, and
    methods when written in the syntax of
    an expression, is allowed.
  calcAge: function (birthYeah) {
      return 2037 - birthYeah;
 }, // note that there is a comma
  // Cleaner syntax! - Using the Keyword:
    this

  calcAge: function () {
      console.log(this);
      return 2037 - this.birthYeah;
 }, // note that there is a comma

  // Use method to add another property
  calcAge: function () {
    this.age = 2037 - this.birthYeah;
    return this.age;
}, //note the comma!

// getSummary()
  getSummary: function () {
    return `${this.firstName} is a
      ${this.calcAge()}-year old
      ${jonas.job}, and he has
      ${this.hasDriversLicense ? 'a' :
      'no'} driver's license.`
    // Note this! Note the ternary
      operator!!

} //note that the last property does not
```

```
112   } //note that the last property does not
  •     need a comma!
113   };
114   // To calculate the age of Jonas
115   console.log(jonas.calcAge());
116   // You can also access it like This
117   console.log(jonas['calcAge']()); //
  •     calcAge has to be a string!
118
119   console.log(jonas.age); // Note: command
  •     D gives you multiple cursors - it means
  •     get next occurence. Do this multiple
  •     times for more occurences.
120   console.log(jonas.age);
121   console.log(jonas.age);
122
123   // Challenge
124   // "Jonas is a 46-year old teacher, and
  •     he has a driver's license"
125   console.log(jonas.getSummary());
126
127   // NOTE: Remember we used methods on
  •     arrays previously. Remember we had
  •     friends.push(). On that array, we
  •     called push(). Arrays ARE ALSO OBJECTS!
  •     They have functions, or in other words,
  •     methods, that can be used to manipulate
  •     them like push, pop, shift and unshift
  •     and many more. But these are built in
  •     objects on arrays.
128
```