Data Structures: Objects

Working with Object Properties and Values

We'll cover the following

- Objects
  - Exercise
- Iterating through object properties
  - Exercise

Objects

A Javascript object is another variable that allows us to store multiple pieces of data.

Like arrays, objects are capable of storing any valid Javascript data type, including arrays and other objects. However, unlike arrays, data in objects are stored in object **properties**.

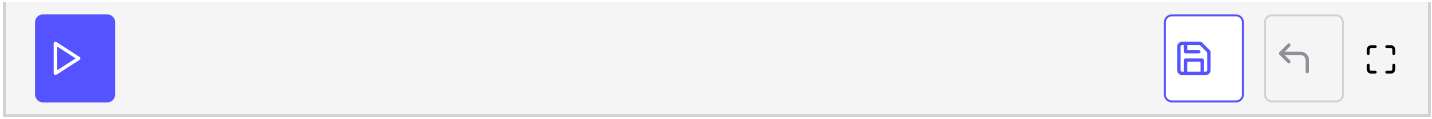
Let's once again take the example of a classroom with students. We will make a single student an *object* with *multiple properties*:

```
var student = {
  name: "Mary",
  age: 10
}
```

An object is declared using curly braces (`{}`). *Properties* and their *values* are stored within the curly braces, separated by a colon (`:`). Each property is separated by a comma (`,`), which comes after each property's *value*.

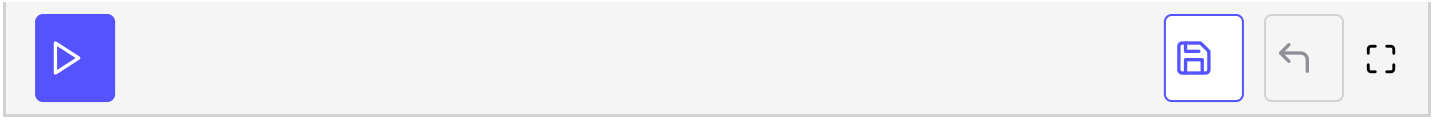
The following object has **two properties**: *name* and *age*. You can access the `student` object's property values by looking up a property using either **dot notation** or **bracket notation**:

```
// accessing values using dot notation
console.log(student.name);
console.log(student.age);

// is the same as accessing values using bracket notation
console.log(student["name"]);
console.log(student["age"]);
```



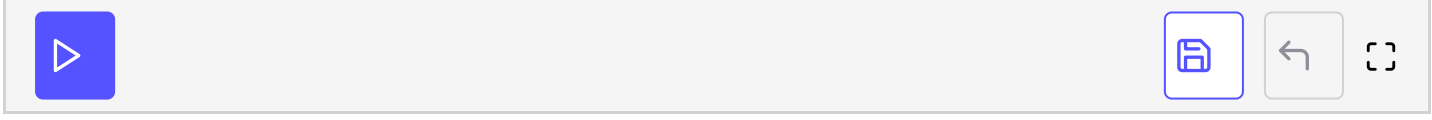
As previously stated, an object's property values can be any valid Javascript data type. Here's a `student` object with a `grades` property with a value that is another object:

```
var student = {
  name: "Mary",
  age: 10,
  grades: {
    quiz1: 90,
    quiz2: 88,
    quiz3: 95
  }
}
```



Once an object has been created, you can add **additional properties** by using *dot* or *bracket* notation:

```
//set new property values using dot notation
student.gender = "female";

//set new property values using bracket notation
student["height"] = "122cm";

console.log(student.gender, student.height);
```



You can also **remove** a property from an object using the `delete` keyword:

```
//remove object properties using the delete keyword
delete student.gender;

console.log(student.gender);
```

Check your Understanding

How would you access the `quiz1` grade in the `student` object shown below?

```
var student = {
  name: "Mary",
  age: 10,
  grades: {
    quiz1: 90,
    quiz2: 88,
    quiz3: 95
  }
}
```

Retake Quiz

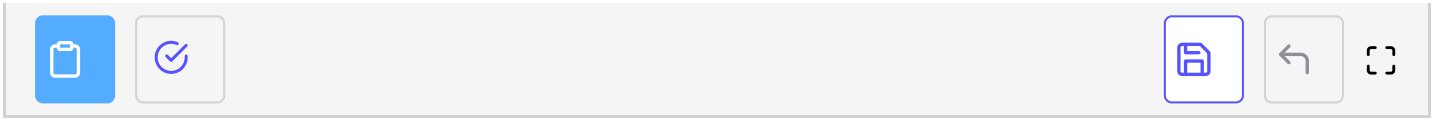
Exercise

Convert the following array of arrays:

```
[ ["Mary", 10], ["Barbara", 11], ["David", 12], ["Alex", 11] ];
```

into an array of *objects*, named `students`. Each object in the array should have a `name` and `age` property.

```
var students = [
  //insert your code here
];
```

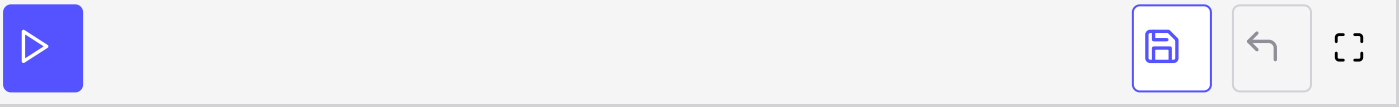


Iterating through object properties

Like with arrays, we can use a loop to go through all an object's properties to access its values.

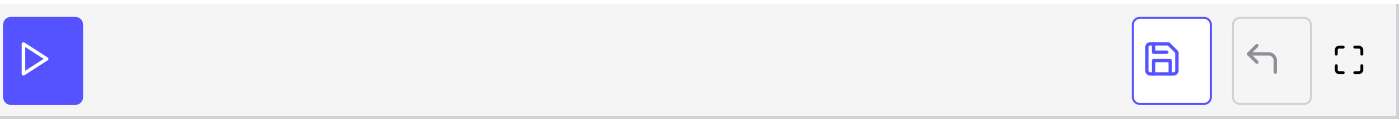
Javascript provides a special `for...in` syntax for iterating through object properties. Let's take a look at another `student` object and go through its properties:

```
var student = {
  name: "Mary",
  age: 10,
  grades: [90, 88, 95]
}

for(property in student) {
  console.log(property);
}
```



Notice how this loop logs the object's **properties** to the console. To access the **value** associated with each property, you must use *bracket* notation, like this:

```
for(property in student) {
  //dot notation does not access property values
  console.log(student.property);

  //bracket notation does
  console.log(student[property]);
}
```



Exercise

Given an array of `student` objects, structured like so:

```
var students = [
  {
    name: "Mary",
    age: 10,
    grades: [90, 88, 95]
  },
  {
    name: "Joseph",
    age: 11,
    grades: [80, 100, 90, 96]
  }
];
```

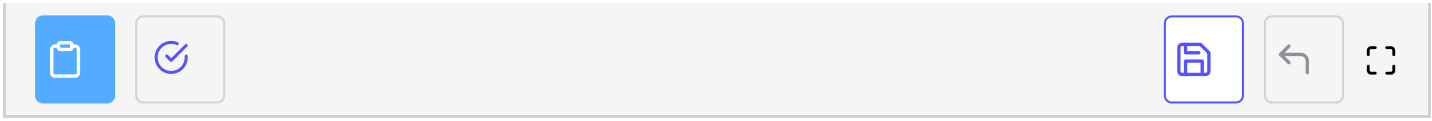
Write a function that **returns an array of grade averages**. For instance, if the returned array was named `averages`:

- `averages[0]` would equal `91`

- `averages[1]` would equal `91.5`

Assume each `student` object in your array has a `grades` property.

```
var getAverages = function(students){
  var averages = [];
  //write your code here
  return averages;
}
```



Now that you have learned about objects, let's explore functions a bit more in the next lesson.