Setting & Deleting Properties

This lesson will teach you how to set properties using dot operators or square brackets, and how to delete properties.

We'll cover the following

Setting Values Using Dot Notation

A property being accessed by the dot operator can be set/updated using the *assignment operator* (`=`).

Example

Let's take a look at an example to see the implementation.

```
//creating an object named employee

var employee = {
  //defining properties of the object
  //setting data values
  name : 'Joe',
  age : 20
}

console.log("Originally age was:",employee.age)
//updating the value of "age"
employee.age = 24
console.log("New age is:",employee.age)
```

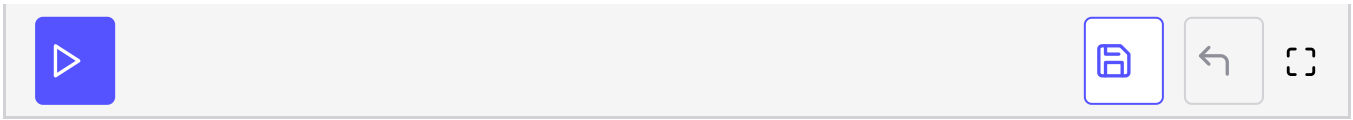
Adding Properties

What if the property doesn't exist? Setting it will automatically create a *new* property.

```
//creating an object named employee

var employee = {
  //defining properties of the object
  //setting data values
  name : 'Joe',
  age : 20
}

console.log("Originally age was:",employee.age)
//updating the value of "age"
employee.age = 24
console.log("New age is:",employee.age)
//creating a new property called designation and setting its value
employee.designation = 'Developer'
console.log("Designation is:",employee.designation)
```



As can be seen from the above example, the property designation was not a part of the `employee` object originally. However, we set its value in **line 15**, which resulted in it being created automatically.

Setting Values Accessed Through Square Brackets

Values accessed through square brackets can also be set/updated using the *assignment operator* (`=`).

Example

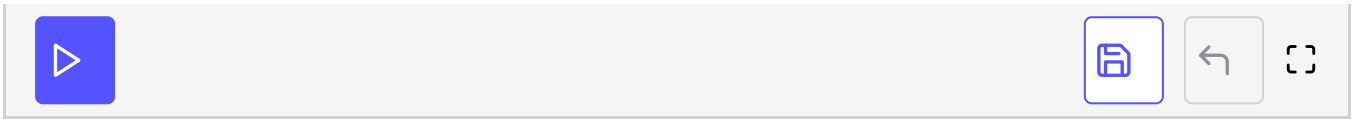
Let's take a look at an example to see the implementation.

```
//creating an object named employee

var employee = {
  //defining properties of the object
  //setting data values
  name : 'Joe',
  age : 20
}

console.log("Originally age was:",employee['age'])
```

```
//updating the value of "age"
employee['age'] = 24
console.log("New age is:",employee['age'])
//creating a new property called designation and setting its value
employee['designation'] = 'Developer'
console.log("Designation is:",employee['designation'])
```



Deleting Properties

Properties can be deleted or removed from an object using the `delete` operator. Both the property and the value will get deleted, therefore, if we want to use the property again, we will have to define it once more.

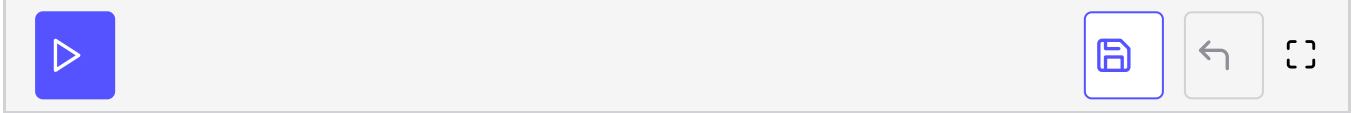
Example

Let's take a look at an example using the `delete` operator.

```
//creating an object named employee

var employee = {
  //defining properties of the object
  //setting data values
  name : 'Joe',
  age : 20
}

//deleting "age" property
delete employee.age
console.log("Age is:",employee.age) //since "age" is deleted "undefined" will be displayed
//deleting "name"
delete employee['name'] //square brackets can also be used
console.log("Name is:",employee.name) //since "name" is deleted "undefined" will be displayed
```

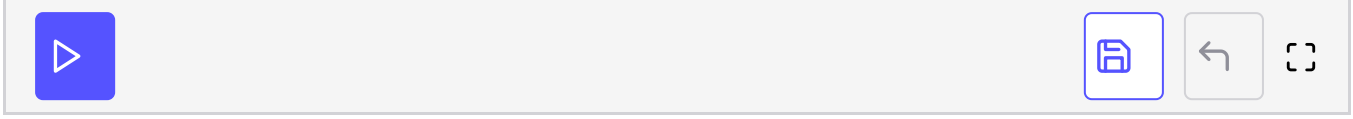


Similarly, you can also delete the function properties of an object.

```
//creating an object named employee

var employee = {
  //defining properties of the object
   display() {
     console.log("Name is Joe")
   }
}
//calling the function
employee.display()
```

```
//deleting the function using delete operator
delete employee.display
employee.display() //error thrown since the function has been deleted
```



> **Important Note:** The `delete` operator has no effect on the variables and functions declared in the global scope. It can only delete object properties and functions.

Now that you know about properties, let's learn about *methods* in objects in the next lesson.