WHAT IS AN OBJECT?

Objects group together a set of variables and functions to create a model of a something you would recognize from the real world. In an object, variables and functions take on new names.

IN AN OBJECT: VARIABLES BECOME KNOWN AS PROPERTIES

If a variable is part of an object, it is called a **property**. Properties tell us about the object, such as the name of a hotel or the number of rooms it has. Each individual hotel might have a different name and a different number of rooms.

IN AN OBJECT: FUNCTIONS BECOME KNOWN AS METHODS

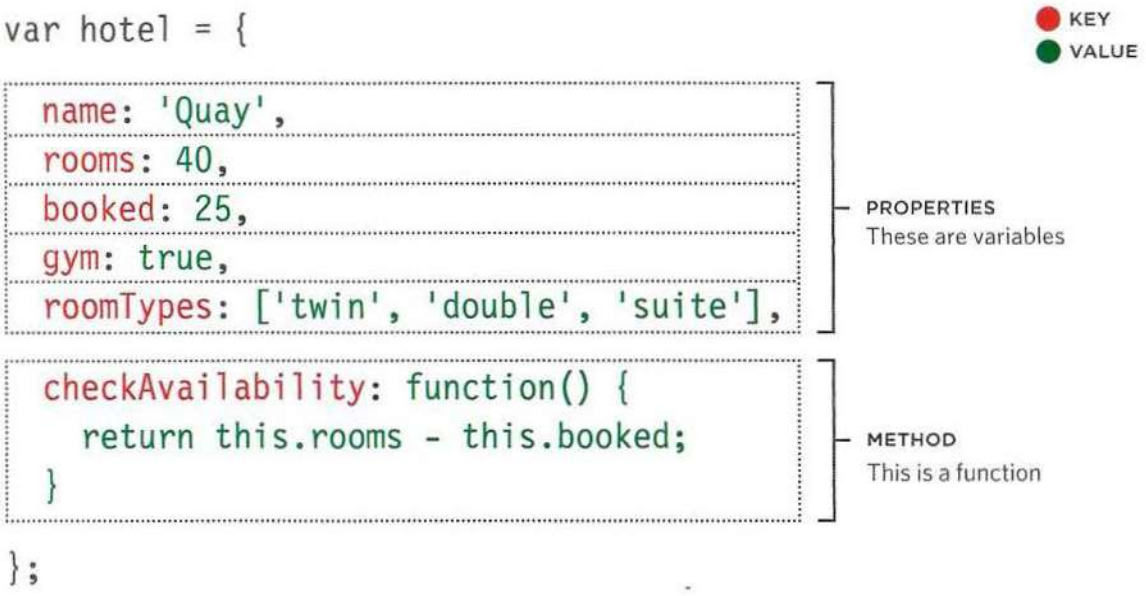
If a function is part of an object, it is called a **method**. Methods represent tasks that are associated with the object. For example, you can check how many rooms are available by subtracting the number of booked rooms from the total number of rooms.

This object represents a hotel. It has five properties and one method. The object is in curly braces. It is stored in a variable called `hotel`.

Like variables and named functions, properties and methods have a name and a value. In an object, that name is called a **key**.

An object cannot have two keys with the same name. This is because keys are used to access their corresponding values.

The value of a property can be a string, number, Boolean, array, or even another object. The value of a method is always a function.



Above you can see a `hotel` object. The object contains the following key/value pairs:

	KEY	VALUE
PROPERTIES:	`name`	string
	`rooms`	number
	`booked`	number
	`gym`	Boolean
	`roomTypes`	array
METHODS:	`checkAvailability`	function

As you will see over the next few pages, this is just one of the ways you can create an object.

Programmers use a lot of name/value pairs:

- HTML uses attribute names and values.
- CSS uses property names and values.

In JavaScript:

- Variables have a name and you can assign them a value of a string, number, or Boolean.
- Arrays have a name and a group of values. (Each item in an array is a name/value pair because it has an index number and a value.)
- Named functions have a name and value that is a set of statements to run if the function is called.
- Objects consist of a set of name/value pairs (but the names are referred to as keys).

CREATING AN OBJECT: LITERAL NOTATION

Literal notation is the easiest and most popular way to create objects. (There are several ways to create objects.)

The object is the curly braces and their contents. The object is stored in a variable called **`hotel`**, so you would refer to it as the **`hotel`** object.

Separate each key from its value using a colon. Separate each property and method with a comma (but not after the last value).

```
var hotel = {
  name: 'Quay',
  rooms: 40,
  booked: 25,
  checkAvailability: function() {
    return this.rooms - this.booked;
  }
};
```

OBJECT
KEY
VALUE

PROPERTIES

METHOD

In the **`checkAvailability()`** method, the **`this`** keyword is used to indicate that it is using the **`rooms`** and **`booked`** properties of ***this*** object.

When setting properties, treat the values like you would do for variables: strings live in quotes and arrays live in square brackets.

ACCESSING AN OBJECT AND DOT NOTATION

You access the properties or methods of an object using dot notation. You can also access properties using square brackets.

To access a property or method of an object you use the name of the object, followed by a period, then the name of the property or method you want to access. This is known as **dot notation**.

The period is known as the **member operator**. The property or method on its right is a member of the object on its left. Here, two variables are created to hold the hotel name and number of vacant rooms.

You can also access the properties of an object (but not its methods) using square bracket syntax.

This time the object name is followed by square brackets, and the property name is inside them.

```
var hotelName = hotel['name'];
```

This notation is used most commonly used when:

- The name of the property is a number (technically allowed, but should generally be avoided)
- A variable is being used in place of the property name (you will see this technique used in Chapter 12)

CREATING OBJECTS USING LITERAL NOTATION

This example starts by creating an object using literal notation.

This object is called `hotel` which represents a hotel called `Quay` with 40 rooms (25 of which have been booked).

Next, the content of the page is updated with data from this object. It shows the name of the hotel by accessing the object's `name` property and the number of vacant rooms using the `checkAvailability()` method.

To access a property of this object, the object name is followed by a dot (the period symbol) and the name of the property that you want.

Similarly, to use the method, you can use the object name followed by the method name. `hotel.checkAvailability()`

If the method needs parameters, you can supply them in the parentheses (just like you can pass arguments to a function).

c3/js/object-literal.js

JAVASCRIPT

```
var hotel = {
  name: 'Quay',
  rooms: 40,
  booked: 25,
  checkAvailability: function() {
    return this.rooms - this.booked;
  }
};

var elName = document.getElementById('hotelName');
elName.textContent = hotel.name;

var elRooms = document.getElementById('rooms');
elRooms.textContent = hotel.checkAvailability();
```

RESULT

CREATING MORE OBJECT LITERALS

JAVASCRIPT

c03/js/object-literal2.js

```
var hotel = {
  name: 'Park',
  rooms: 120,
  booked: 77,
  checkAvailability: function() {
    return this.rooms - this.booked;
  }
};

var elName = document.getElementById('hotelName');
elName.textContent = hotel.name;

var elRooms = document.getElementById('rooms');
elRooms.textContent = hotel.checkAvailability();
```

RESULT

Here you can see another object. Again it is called `hotel`, but this time the model represents a different hotel. For a moment, imagine that this is a different page of the same travel website.

The Park hotel is larger. It has 120 rooms and 77 of them are booked.

The only things changing in the code are the values of the `hotel` object's properties:

- The name of the hotel
- How many rooms it has
- How many rooms are booked

The rest of the page works in exactly the same way. The name is shown using the same code. The `checkAvailability()` method has not changed and is called in the same way.

If this site had 1,000 hotels, the only thing that would need to change would be the three properties of this object. Because we created a model for the hotel using data, the same code can access and display the details for any hotel that follows the same data model.

If you had two objects on the same page, you would create each one using the same notation but store them in variables with different names.