Objects group together a set of variables and functions to create a model of something you would recognize from the real world.

Object properties + methods

In an object:

1. **Variables** become known as **properties**
2. **Functions** become known as **methods**

Properties tell us about the object, such as the height of a chair and how many chairs there are -

```
function Chair( height, number ){
    this.h = height;
    this.count = number;

}
```

Methods represent tasks that are associated with the object, e.g. count the height of all chairs by adding all heights together

Properties & objects

```
var hotel = {
  // the following are key: value pairs.
    name: "Radisson",
    rooms: 55,
    booked: 15,
    spa: false,
    roomTypes: ["single", "twin", "suite"],

    checkAvailability: function( ) {
      return this.rooms - this.booked;
    }
}
```

Properties

Method

name, rooms, booked - are **keys** in the object
Object cannot have two keys with the same name
Value can be anything you want (string, number, boolean, function, another object)

Creating an object: literal notation

```
var hotel = {
  // the following are key: value pairs.
    name: "Radisson",
    rooms: 55,
    booked: 15,
    spa: false,
    roomTypes: ["single", "twin", "suite"],

    checkAvailability: function( ) {
      return this.rooms - this.booked;
    }
}
```

Object is stored in a variable called hotel.

Each **key** is separated from its **value** by a colon :

Each property and method is separated by a comma, this keyword indicates that "rooms" and "booked" properties should be taken from this object

Accessing an object

There are two ways to access properties in an object 1) dot notation 2) square brackets

```
var hotel = {
  // the following are key: value pairs.
    name: "Radisson",
    rooms: 55,
    booked: 15,
    spa: false,
    roomTypes: ["single", "twin", "suite"],

    checkAvailability: function( ) {
      return this.rooms - this.booked;
    }
}
```

Properties

Method

dot notation

```
var hotelName = hotel.name;
var roomsAvailable = hotel.checkAvailabilty( );
```

square brackets

```
var hotelName = hotel["name"];
var roomsAvailable = hotel["checkAvailability"]( );
```

Creating an object: constructor notation

```
var hotel = {
  // the following are key: value pairs.
    name: "Radisson",
    rooms: 55,
    booked: 15,
    spa: false,
    roomTypes: ["single", "twin", "suite"],

    checkAvailability: function( ) {
      return this.rooms - this.booked;
    }
}
```

The new keyword creates a blank object
Then you can add properties and method using dot notation

Updating an object

```
var hotel = {
  // the following are key: value pairs.
    name: "Radisson",

}
```

dot notation

```
hotel.name = "Hilton";
```

square brackets

```
hotel["name"] = "Hilton";
```

Deleting property from an object

```
var hotel = {
  // the following are key: value pairs.
    name: "Radisson",

}
```

dot notation

```
    delete hotel.name;
```

Creating many objects: constructor notation

Often - you will want several objects to represent similar thing
You can use function as a template to create several objects
First we need to create template with object's properties and methods

```
var theHotel (theName, theRooms, theBooked) = {
  // the following are key: value pairs.
    this.name: theName,
    this.rooms: theRooms,
    this.booked: theBooked,

    checkAvailability: function( ) {
      return this.rooms - this.booked;
    }
}

var radissonHotel = new theHotel("Radisson", 55, 15);
var hiltonHotel = new theHotel("Hilton", 60, 60);
```

Properties

Method