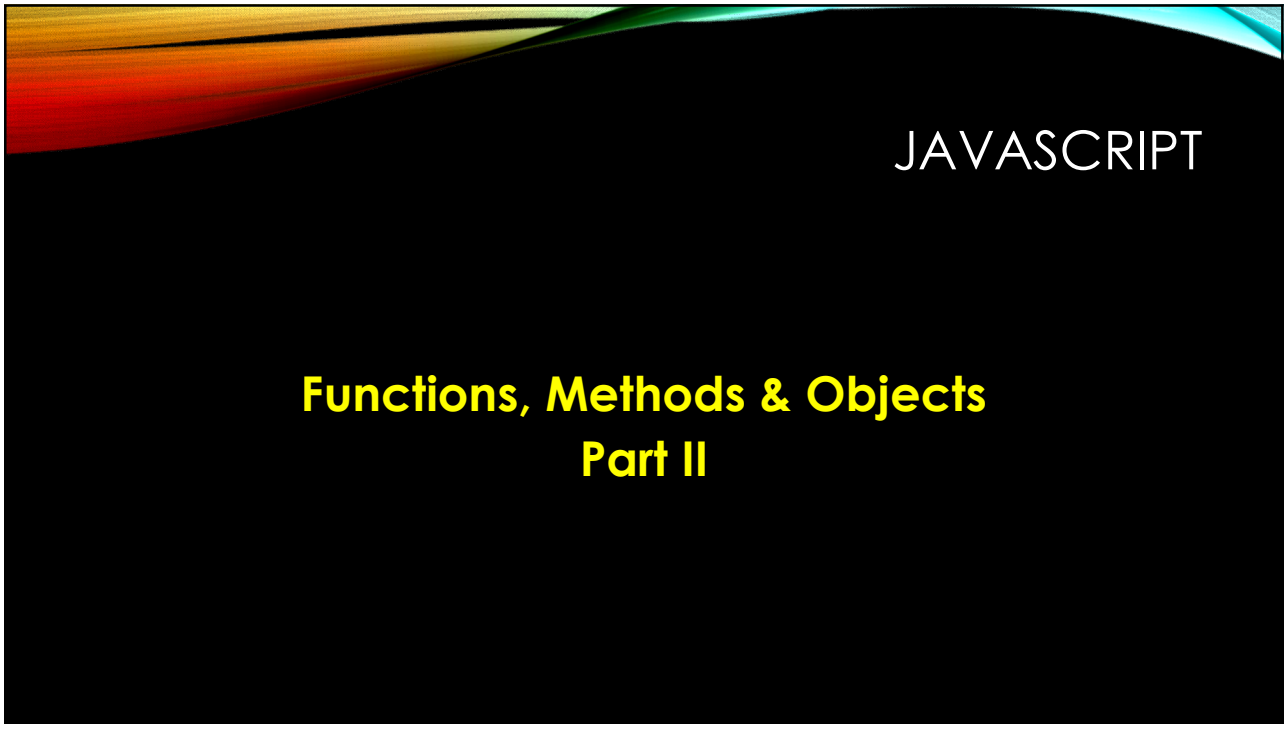

JAVASCRIPT
Haider M. Habeeb
University of Babylon
College of Information Technology
2009

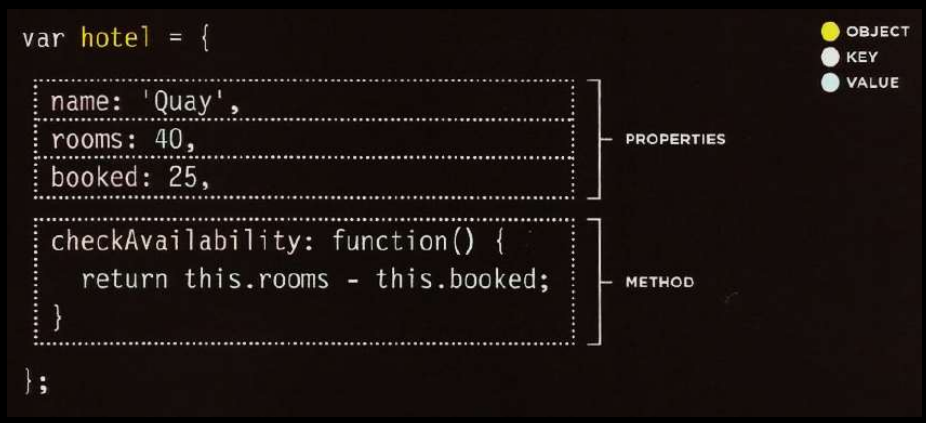

JAVASCRIPT
Functions, Methods & Objects
Part II

OBJECT

- There are two ways for creating Object:
  1. **Literal Notation**
  2. **Constructor Notation**

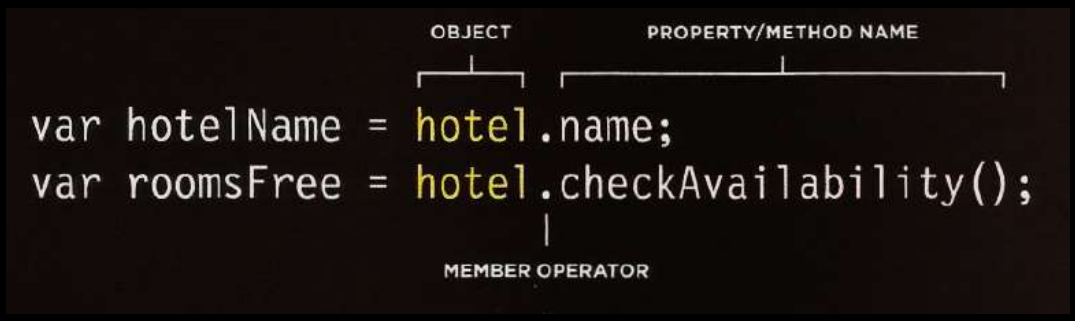
OBJECT

- Literal Notation



OBJECT

- **Accessing an Object:**
- You can access the properties and methods of an object using dot notation



OBJECT

- **Accessing an Object:**
- You can also access properties using square brackets

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

OBJECT

- Creating objects using literal notation:

```
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();
```

OBJECT

- Constructor Notation

```
var hotel = new Object();

hotel.name = 'Quay';
hotel.rooms = 40;
hotel.booked = 25;

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

OBJECT
KEY
VALUE

PROPERTIES

METHOD

OBJECT

- Updating an Object:

```
hotel.name = 'Park';
```

```
hotel['name'] = 'Park';
```

```
delete hotel.name;
```

```
hotel.name = '';
```

OBJECT

- Creating Many Objects:
  - Sometimes you will want several objects to represent similar things.
  - Object constructors can use a function as a **template** for creating objects.
  - First, create the template with the object's properties and methods.

OBJECT

- Constructor Function
  - The ***this*** keyword is used instead of the object name to indicate that the property or method belongs to the object that ***this*** function creates.

```
function Hotel(name, rooms, booked) {
  this.name = name;
  this.rooms = rooms;
  this.booked = booked;

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

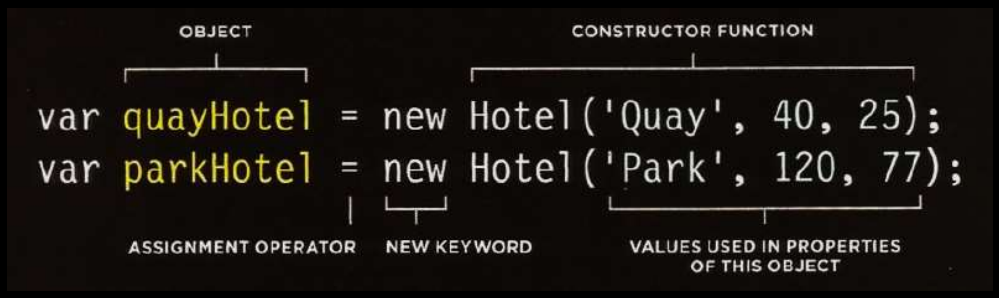
OBJECT

- Constructor Function

- Each statement that creates a new property or method for this object *ends in a semicolon* (not a comma, which is used in the literal syntax).
- The name of a constructor function usually begins with a *capital letter* (unlike other functions, which tend to begin with a lowercase character).
- The uppercase letter is supposed to help remind developers to use the *new* keyword when they create an object using that function.

OBJECT

- Creating instances of the object using the constructor function.
- The *new* keyword followed by a call to the function creates new object.
- The properties of each object are given as *arguments* to the function.



OBJECT

- Creating objects using constructor syntax

```
var hotel = new Object();

hotel.name = 'Park';
hotel.rooms = 120;
hotel.booked = 77;
hotel.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();
```

OBJECT

- Create & access objects constructor notation

```
function Hotel(name, rooms, booked) {
  this.name = name;
  this.rooms = rooms;
  this.booked = booked;
  this.checkAvailability = function() {
    return this.rooms - this.booked;
  };
}
```

```
var quayHotel = new Hotel('Quay', 40, 25);
var parkHotel = new Hotel('Park', 120, 77);

var details1 = quayHotel.name + ' rooms: ';
    details1 += quayHotel.checkAvailability();
var elHotel1 = document.getElementById('hotel1');
elHotel1.textContent = details1;

var details2 = parkHotel.name + ' rooms: ';
    details2 += parkHotel.checkAvailability();
var elHotel2 = document.getElementById('hotel2');
elHotel2.textContent = details2;
```

OBJECT

- Adding and removing properties

```
var hotel = {
  name : 'Park',
  rooms : 120,
  booked : 77,
};

hotel.gym = true;
hotel.pool = false;
delete hotel.booked;
```

- Recap: Ways to create objects

LITERAL NOTATION

```
var hotel = {}

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

OBJECT CONSTRUCTOR NOTATION

```
var hotel = new Object();

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

OBJECT

LITERAL NOTATION

A colon separates the key/value pairs.
There is a comma between each key/value pair.

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

OBJECT CONSTRUCTOR NOTATION

The function can be used to create multiple objects.
The this keyword is used instead of the object name.

```
function Hotel(name, rooms, booked) {
  this.name = name;
  this.rooms = rooms;
  this.booked = booked;
  this.checkAvailability = function() {
    return this.rooms - this.booked;
  };
}
var quayHotel = new Hotel('Quay', 40, 25);
var parkHotel = new Hotel('Park', 120, 77);
```

OBJECT

- Arrays are objects:
- Arrays are a special type of objects

AN OBJECT

PROPERTY:	VALUE:
room1	420
room2	460
room3	230
room4	620

AN ARRAY

INDEX NUMBER:	VALUE:
0	420
1	460
2	230
3	620

OBJECT

- Arrays of Objects & Objects in Arrays:
  - You can combine arrays and objects to create complex data structures.
  - Arrays can store a series of objects (and remember their order)
  - Objects can hold arrays (as values of their properties)

OBJECT

ARRAYS IN AN OBJECT

The property of any object can hold an array. On the left, each item on a hotel bill is stored separately in an array. To access the first charge for `room1` you would use:

```
costs.room1.items[0];
```

PROPERTY:	VALUE:
room1	items[420, 40, 10]
room2	items[460, 20, 20]
room3	items[230, 0, 0]
room4	items[620, 150, 60]

OBJECT

OBJECTS IN AN ARRAY

The value of any element in an array can be an object (written using the object literal syntax). Here, to access the phone charge for room three, you would use:

```
costs[2].phone;
```

INDEX NUMBER:	VALUE:
0	{accom: 420, food: 40, phone: 10}
1	{accom: 460, food: 20, phone: 20}
2	{accom: 230, food: 0, phone: 0}
3	{accom: 620, food: 150, phone: 60}

OBJECT

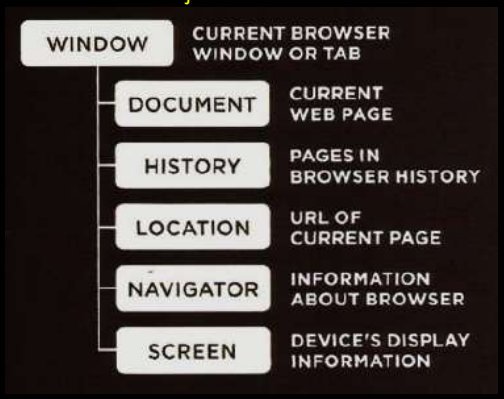
- What are built-in objects?
  - Browsers come with a set of built-in objects that act like a toolkit for creating interactive web pages.
  - The objects ***you*** create will usually be specifically written to suit ***your needs***.
  - Built-in objects contain functionality commonly needed by many scripts.
  - Built-in objects help you get a wide range of information such as the width of the browser window, or the length of text a user entered into a form field.
  - You access their properties or methods using dot notation, just like you would access the properties or methods of an object you had written yourself.

OBJECT

- Three groups of Built-in Objects:
  - Browser Object Model:
    - Contains objects that represent the current browser window or tab. It contains objects that model things like browser history and the device's screen.
  - Document Object Model:
    - Uses objects to create a representation of the current page. It creates a new object for each element (and each individual section of text) within the page.
  - Global JavaScript Objects:
    - Represent things that the JavaScript language needs to create a model of. For example, there is an object that deals only with dates and times.

OBJECT

- Browser Object Model:



EXAMPLES

The `window` object's `print()` method will cause the browser's print dialog box to be shown:

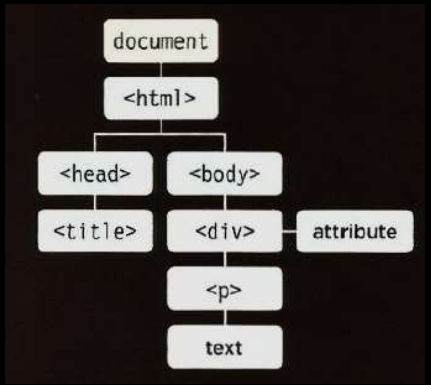
```
window.print();
```

The `screen` object's `width` property will let you find the width of the device's screen in pixels:

```
window.screen.width;
```

OBJECT

- Document Object Model:



EXAMPLES

The `document` object's `getElementById()` method gets an element by the value of its `id` attribute:

```
document.getElementById('one');
```

The `document` object's `lastModified` property will tell you the date that the page was last updated:

```
document.lastModified;
```

OBJECT

- Global JavaScript Objects:

STRING	FOR WORKING WITH STRING VALUES
NUMBER	FOR WORKING WITH NUMERIC VALUES
BOOLEAN	FOR WORKING WITH BOOLEAN VALUES
DATE	TO REPRESENT AND HANDLE DATES
MATH	FOR WORKING WITH NUMBERS AND CALCULATIONS
REGEX	FOR MATCHING PATTERNS WITHIN STRINGS OF TEXT

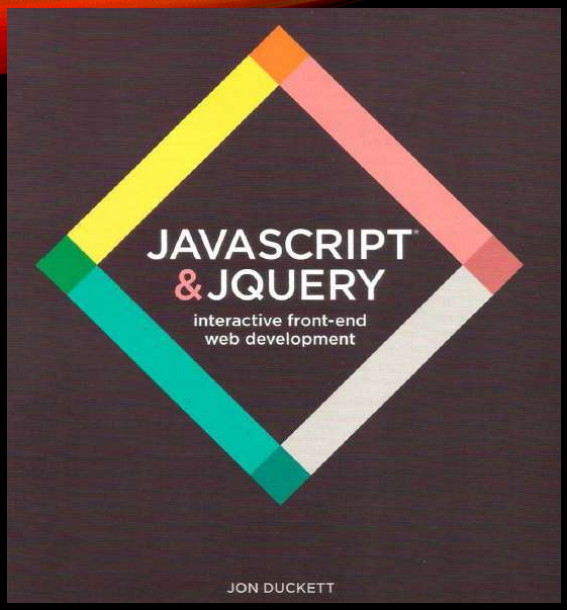
EXAMPLES

The `String` object's `toUpperCase()` method makes all letters in the following variable uppercase:

```
hotel.toUpperCase();
```

The `Math` object's `PI` property will return the value of pi:

```
Math.PI();
```



OBJECT

- Dear Students:
- You are required reading the part of built-in objects, page number 120 until the end of chapter Three.
- You will find most of built-in objects with examples.


Thank you!