Objects

JavaScript Objects

JavaScript variables are containers for data values. **Objects** are variables too, but they can contain many values.

Think of an object as a list of values that are written as **name:value** pairs, with the names and the values separated by colons.
Example:

```
var person = {
 name: "John", age: 31,
 favColor: "green", height: 183
};
```

These values are called **properties**.

Property	Property Value
name	John
age	31
favColor	green
height	183

JavaScript objects are containers for **named values**.

Object Properties

You can access object properties in two ways.

```
objectName.propertyName
//or
objectName['propertyName']
```

This example demonstrates how to access the age of our person object.

```
var person = {
 name: "John", age: 31,
 favColor: "green", height: 183
};
var x = person.age;
var y = person['age'];
```

Try It Yourself

JavaScript's built-in **length** property is used to count the number of characters in a property or string.

```
var course = {name: "JS", lessons: 41};
document.write(course.name.length);
//Outputs 2
```

Try It Yourself

Objects are one of the core concepts in JavaScript.

Object Methods

An object **method** is a property that contains a **function definition**.

Use the following syntax to access an object method.

```
objectName.methodName()
```

As you already know, **document.write()** outputs data. The **write()** function is actually a method of the **document** object.

```
document.write("This is some text");
```

Try It Yourself

Methods are functions that are stored as object properties.

The Object Constructor

In the previous lesson, we created an object using the **object literal** (or initializer) syntax.

```
var person = {
name: "John", age: 42, favColor: "green"
};
```

This allows you to create only a single object.
Sometimes, we need to set an "**object type**" that can be used to create a number of objects of a single type.
The standard way to create an "object type" is to use an object **constructor function**.

```
function person(name, age, color) {
  this.name = name;
  this.age = age;
  this.favColor = color;
}
```

The above function (person) is an object constructor, which takes parameters and assigns them to the object properties.

The **this** keyword refers to the **current object**.
Note that **this** is not a variable. It is a keyword, and its value cannot be changed.

Creating Objects

Once you have an object constructor, you can use the **new** keyword to create new objects of the same type.

```
var p1 = new person("John", 42, "green");
var p2 = new person("Amy", 21, "red");

document.write(p1.age); // Outputs 42
document.write(p2.name); // Outputs "Amy"
```

Try It Yourself

p1 and *p2* are now objects of the **person** type. Their properties are assigned to the corresponding values.

Creating Objects

Consider the following example.

```
function person (name, age) {
  this.name = name;
  this.age = age;
}
var John = new person("John", 25);
var James = new person("James", 21);
```

Try It Yourself

Access the object's properties by using the **dot syntax**, as you did before.

Object's Name	Property's name
John .	name
John .	age
James .	name
James .	age

Understanding the creation of objects is essential.

Object Initialization

Use the object **literal** or **initializer** syntax to create single objects.

```
var John = {name: "John", age: 25};
var James = {name: "James", age: 21};
```

Objects consist of properties, which are used to describe an object. Values of object properties can either contain primitive data types or other objects.

Using Object Initializers

Spaces and line breaks are not important. An object definition can span multiple lines.

```
var John = {
  name: "John",
  age: 25
};
var James = {
  name: "James",
  age: 21
};
```

No matter how the object is created, the syntax for accessing the properties and methods does not change.

```
document.write(John.age);
//Outputs 25
```

Try It Yourself

Don't forget about the second accessing syntax: John['age'].

Methods

Methods are functions that are stored as object properties.

Use the following syntax to create an object method:

```
methodName = function() { code lines }
```

Access an object method using the following syntax:

```
objectName.methodName()
```

A method is a function, belonging to an object. It can be referenced using the **this** keyword. The **this** keyword is used as a reference to the current object, meaning that you can access the objects properties and methods using it.

Defining methods is done inside the constructor function.

For Example:

```
function person(name, age) {
  this.name = name;
  this.age = age;
  this.changeName = function (name) {
    this.name = name;
  }
}
```

```
var p = new person("David", 21);
p.changeName("John");
//Now p.name equals to "John"
```

Try It Yourself

In the example above, we have defined a method named **changeName** for our person, which is a function, that takes a parameter **name** and assigns it to the **name** property of the object. **this.name** refers to the name property of the object.

> The **changeName** method changes the object's **name** property to its argument.

Methods

You can also define the function outside of the constructor function and associate it with the object.

```
function person(name, age) {
  this.name= name;
  this.age = age;
  this.yearOfBirth = bornYear;
}
function bornYear() {
  return 2016 - this.age;
}
```

As you can see, we have assigned the object's **yearOfBirth** property to the **bornYear** function. The **this** keyword is used to access the *age* property of the object, which is going to call the method.

> Note that it's not necessary to write the function's parentheses when assigning it to an object.

Methods

Call the method as usual.

```
function person(name, age) {
  this.name= name;
  this.age = age;
  this.yearOfBirth = bornYear;
}
function bornYear() {
  return 2016 - this.age;
}

var p = new person("A", 22);
document.write(p.yearOfBirth());
// Outputs 1994
```

Try It Yourself

> Call the method by the **property name** you specified in the constructor function, rather than the function name.

END.