

JavaScript and Objects

This lesson explains the basic concepts of Object how do we create an object in JavaScript.

WE'LL COVER THE FOLLOWING



- What's an Object?
- What does this have to do with code?
- JavaScript and Objects
- Creating an Object
- Accessing an Object's Properties
- Modifying an Object

What's an Object?

Think about objects in the non-programming sense, like a pen. A pen can have different ink colors, be manufactured by different people, have a different tip, and many other properties.

Similarly, an *object* in programming is an *entity that has properties*. Each property defines a characteristic of the object. A property can be a piece of information associated with the object (the color of the pen) or an action (the pen's ability to write).

What does this have to do with code?

Object-oriented programming (OOP for short) is a way to write programs using objects. When using OOP, you write, create, and modify objects, and the objects make up your program.

OOP changes the way a program is written and organized. So far, you've been writing function-based code, sometimes called [procedural programming](#). Now

let's discover how to write object-oriented code.

JavaScript and Objects

Like many other languages, JavaScript supports programming with objects. It provides a number of predefined objects while also letting you create your own.

Creating an Object

Here is the JavaScript representation of a blue Bic ballpoint pen.

```
const pen = {  
  type: "ballpoint",  
  color: "blue",  
  brand: "Bic"  
};
```



As stated earlier, a JavaScript object can be created by simply setting its properties within a pair of curly braces: `{...}`. Each property is a key/value pair. This is called an *object literal*.

The semicolon `;` after the closing brace is optional, but it's safer to add it anyway.

The above code defines a variable named `pen` whose value is an object: you can therefore say `pen` is an object. This object has three properties: `type`, `color` and `brand`. Each property has a name and a value and is followed by a comma `,` (except the last one).

Accessing an Object's Properties

After creating an object, you can access the value of its properties using *dot notation* such as `myObject.myProperty`.

```
const pen = {
  type: "ballpoint",
  color: "blue",
  brand: "Bic"
};

console.log(pen.type); // "ballpoint"
console.log(pen.color); // "blue"
console.log(pen.brand); // "Bic"
```



Accessing an object's property is an expression that produces a value. Such an *expression* can be included in more complex ones. For example, here's how to show our pen properties in one statement.

```
const pen = {
  type: "ballpoint",
  color: "blue",
  brand: "Bic"
};

console.log(`I write with a ${pen.color} ${pen.brand} ${pen.type} pen`);
```



Modifying an Object

Once an object is created, you can change the value of its properties with the syntax `myObject.myProperty = newValue`.

```
const pen = {
  type: "ballpoint",
  color: "blue",
  brand: "Bic"
};

pen.color = "red"; // Modify the pen color property

console.log(`I write with a ${pen.color} ${pen.brand} ${pen.type} pen`);
```



JavaScript even offers the ability to dynamically add new properties to an already created object.

```
const pen = {  
  type: "ballpoint",  
  color: "blue",  
  brand: "Bic"  
};
```



```
pen.price = "2.5"; // Set the pen price property
```

```
console.log(`My pen costs ${pen.price}`);
```

