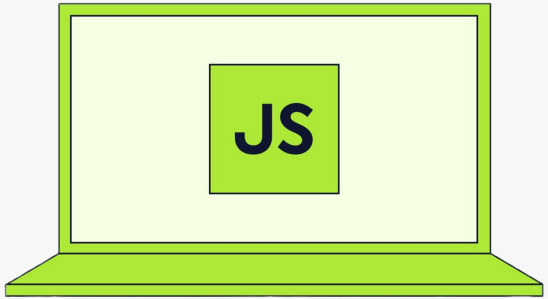
Newton School of Technology

The Complete Javascript Course



@newtonschool

Lecture 7: Introduction to Objects

-Vishal Sharma

JS

Table of Contents

Newton School of Technology

- Creating Objects
  - What are objects? Real-world analogies
  - Why use Object vs Other Data Types
  - Objects as key value pairs
- Accessing Properties
  - Dot Notation Deep Dive
  - Bracket Notation in Depth
- Modifying Objects
  - Adding Properties
  - Updating Properties
  - Deleting Properties
- Iterate over an object using for...in loop
- Iterate over an object using for...of loop

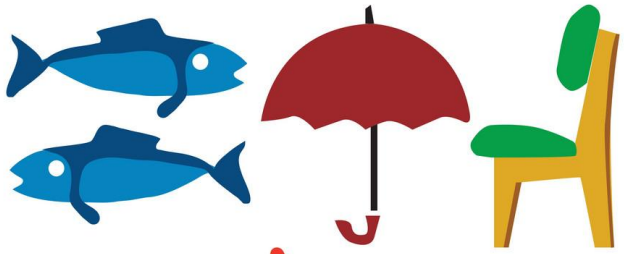
Newton School of Technology

Creating Objects

Objects in real world

Newton School of Technology

In the real world, **objects** are entities or things that have distinct characteristics and behaviors. Objects can be described by their **properties** (attributes) and can perform **actions** (methods).



Each of these objects has some physical properties as well as some functionality. Can you name a few for each of them?

Objects in real world: Example

Newton School of Technology

Let’s have a look at this car. It has certain properties and some functions/methods it can perform.

Properties:-

1. color: red
2. make: Toyota
3. year: 2023
4. etc.

Methods/Functions:-

1. start
2. drive
3. etc.

Objects in Programming

Newton School of Technology

They are collections of related data and functionality grouped together, often mirroring real-world entities. Let's write an object from the car from the previous slide.

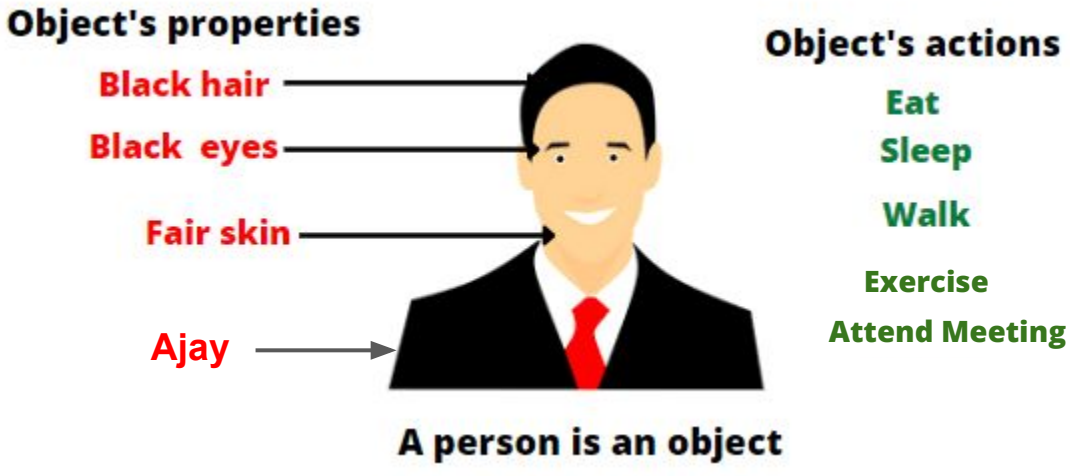
```
const car = {
    // Properties
    color: "red",
    make: "Toyota"
    year: 2023

    // Methods
    start: function() {
      console.log("The car has started.");
    },
    drive: function() {
      console.log("The car is driving.");
    }
};
```

In JavaScript, methods are also considered properties of an object.

Person as an Object

Newton School of Technology

Similarly, we can create an object to represent a person. Imagine Ajay, a busy professional juggling multiple tasks in a day. Alex has distinct traits like a name, black hair, fair skin, etc along with several actions and functions it can perform.



Can you transform this description into an object? Function might only contain console.log() for now?

The Persona of Ajay: A Living Object

Newton School of Technology

Here your implementation might differ; I have only taken a few actions as object methods.

```
const ajay = {
    // Properties
    name: "Ajay",
    hairColor: "black",
    skinTone: "fair",

    // Methods
    attendMeeting: function() {
        console.log(`${ajay.name} is attending a meeting.`);
    },
    exercise: function() {
        console.log(`${ajay.name} is cycling to stay fit.`);
    }
};

// Example Usage
ajay.attendMeeting();
console.log(ajay.name);
```

Now you should feel confident writing an object by yourself.

Object Creation Workshop

Workshop: Question

Create an object to represent a smartphone with the following properties: brand, model, price, and features (as an array). And write some methods like messaging, calling, etc.

Be confident, and you will be able to write in one go.

Newton School of Technology

Objects vs Other Data Types

Newton School of Technology

Objects vs Primitives: A Unified Concept

Objects are derived from primitive types but differ in storing and managing data, grouping related data and behaviors for more flexibility.

```
const ajay = {
    // Primitive Data Types as Properties
    name: "Ajay",           // String (Primitive)
    age: 25,                // Number (Primitive)
    isEmployed: true,       // Boolean (Primitive)
    address: null,          // Null (Primitive)
    gender: undefined,      // Undefined (Primitive)

    // Methods related to person
    greet: function(){
        console.log(`Hello, my name is ${ajay.name}`);
    },
    updateEmploymentStatus: function(status){
        this.isEmployed = status;
        console.log(`Employment: ${ajay.isEmployed}`);
    }
}
```

Notice how various data types are included inside the object named 'ajay'.

How objects are different from primitives

Newton School of Technology

They differ in the following ways:-

Aspect	Primitives	Objects
Structure	Simple, single values like numbers or text.	Complex things that store many related values or actions.
Storage	Small and fast.	Bigger and slower.
Access	Accessed by identifier storing value.	Accessed via properties or methods (e.g., object.property).
Behavior	No inherent behavior; only stores data.	Can include methods for processing data.
Example	var car = “Toyota”	Visit slide 6 to see the example.

We will discuss access methods in later slides

Newton School of Technology

Objects as Key-Value Pairs

Objects as key-value pairs

Newton School of Technology

Objects are essentially collections of related information represented as **key-value pairs**. Values stored are accessible via that key.

key	value
name	Ajay
age	30
occupation	Engineer

```
let person = {
    name: "Ajay",
    age: 30,
    occupation: "Engineer"
};

console.log(person.name); // Output: "Ajay"
console.log(person.age);  // Output: 30
```

Objects as key-value pairs

Newton School of Technology

Methods also need to be accessed via its key as it is also considered as property.

```
let person = {
    name: "Ajay",
    greet: function() {
        console.log("Hello, ", + person.name);
    }
};

person.greet(); // Output: Hello, Ajay
```

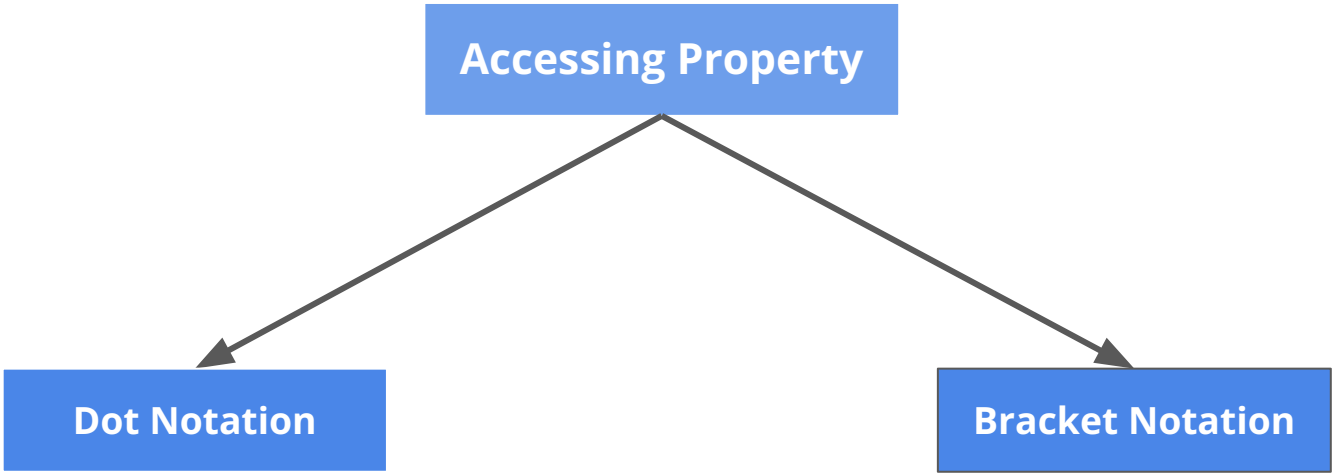
Newton School of Technology

Accessing Properties

Different ways to access properties

Newton School of Technology

There are two main ways to access the properties of an object



Accessing Property: Dot Notation

Newton School of Technology

In JavaScript, dot notation is the simplest and most common way to access the properties of an object. You use a dot (.) followed by the property name.

```
object.property
```

- object is the object you're working with.
- property is the name of the property you want to access.

Dot Notation: example

Newton School of Technology

Let's revisit the car object with the properties `color`, `make`, and `year`, and then access those properties using dot notation.

```
let car = {
    color: "Red",
    make: "Toyota",
    year: 2023
};

console.log(car.color);  // Output: Red
console.log(car.make);   // Output: Toyota
console.log(car.year);   // Output: 2023
```

Accessing properties using dot notation

Dot Notation: example

Newton School of Technology

Accessing the methods are no different.

```
let car = {
    color: "Red",
    make: "Toyota",
    year: 2023,
    displayInfo: function() {
        console.log(`This is a ${car.year} ${car.make} ${car.color} car.`);
    }
};

// Accessing and calling the method using dot notation
car.displayInfo();
// Output: this is a 2023 Toyota Red car.
```

Accessing Property: Bracket Notation

Newton School of Technology

Bracket notation is used to access both properties and methods of an object, especially when the property names are dynamic, contain spaces, or are stored in variables.

```
object[property]
```

Use single or double inverted commas in case property is a string.

Bracket Notation: example

Newton School of Technology

Let's reuse the car example, object creation part remains the same but we will use brackets instead of dot to access the object property.

```
// Accessing properties using bracket notation
console.log(car["color"]);  // Output: Red
console.log(car["make"]);   // Output: Toyota
console.log(car["year"]);   // Output: 2023

// Accessing and calling the method using bracket notation
car["displayInfo"]();
// Output: This is a 2023 Toyota Red car.
```

This notation is very useful in case property name is dynamic i.e. changes frequently maybe based on some condition.

Bracket Notation: example

Newton School of Technology

Accessing the methods are no different.

```
// Let's say the property name is dynamic, based on a condition
let propertyName = "make";
// This can change based on some condition

// Accessing the property dynamically using bracket notation
console.log(car[propertyName]);
// Output: Toyota

// Changing the condition
propertyName = "year";

// Accessing another property based on the updated condition
console.log(car[propertyName]);  // Output: 2023
```

Here we are using propertyName to dynamically access different properties of the car object.

Newton School of Technology

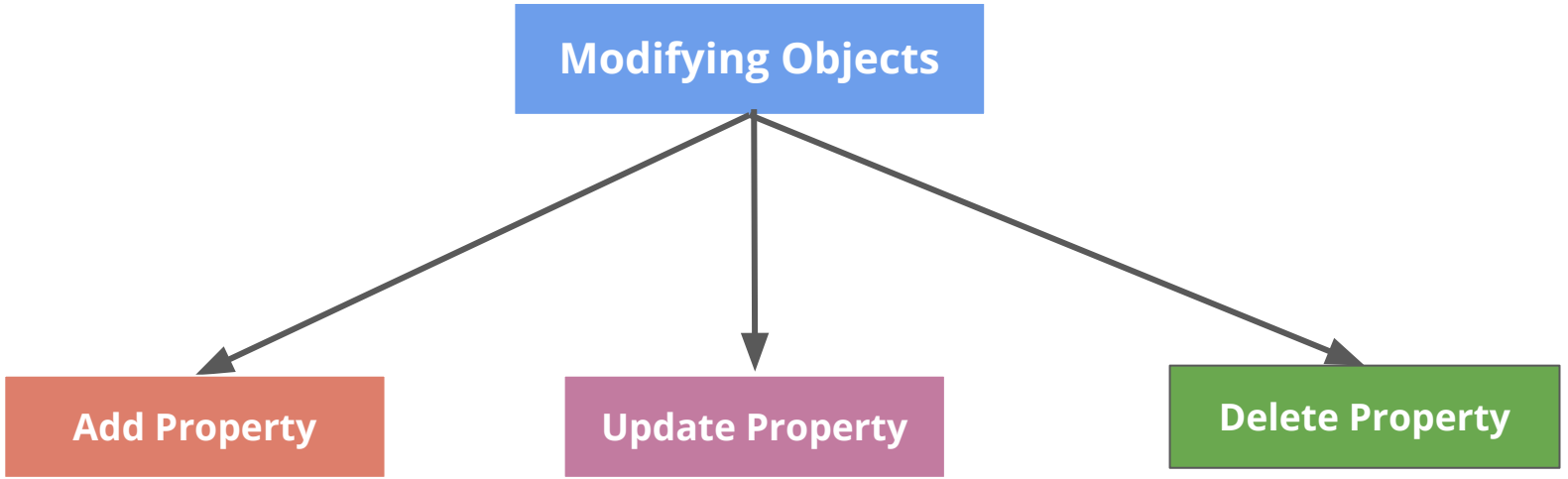
Modifying Objects

Why Modify Objects?

Newton School of Technology

In JavaScript, objects often need to change to reflect new information.

We'll explore three ways to modify objects:



How to add properties

Newton School of Technology

You can add new properties to an object anytime, expanding it dynamically. For example, if you forget to define the car color, you can add it later like this:

```
let car = { make: "Toyota", year: 2023 };

car.color = "Red";  // Adding a new property

console.log(car.color);  // Output: Red
```

We added a new property using the dot operator and assigned it the value "Red".

Adding a method property

Newton School of Technology

Here we have added a method property using dot operator.

```
let car = {
    make: "Toyota",
    year: 2023
};

// Adding a method using dot notation
car.startEngine = function() {
    console.log("The engine has started.");
};

// Calling the method
car.startEngine();
// Output: The engine has started.
```

Try adding one more method property by yourself.

How to Update Properties

Newton School of Technology

You can modify the value of an existing property just the way you create a new property.

```
let car = {
    make: "Toyota",
    year: 2023,
    color: "Red"
};

car.color = "Blue";
// Updating an existing property

console.log(car.color);
// Output: Blue
```

Earlier color was ‘Red’, we have later changed the value to ‘Blue’ by using assignment operator.

How to Update Properties

Newton School of Technology

In the similar fashion we can update method properties too.

```
let car = {
    make: "Toyota",
    year: 2023,
    startEngine: function() {
      console.log("The engine has started.");
    }
};

// Updating the startEngine method
car.startEngine = function() {
    console.log("The engine starts with a roar!");
};

// Calling the updated method
car.startEngine();
// Output: The engine starts with a roar!
```

Sounds simple right??

How to Delete Properties

Newton School of Technology

You can remove properties from objects using the `delete` keyword.

```
let car = {
    make: "Toyota",
    year: 2023,
    color: "Red"
};

delete car.color;
// Deleting a property

console.log(car.color);
// Output: undefined
```

We got `undefined` because the property was deleted.

Caution

- Deleting a property makes it completely unavailable in the object.
- The deleted property cannot be recovered, but the object itself remains intact.

How to Delete Properties

Newton School of Technology

Methods can be deleted from an object just like properties using the `delete` operator.

```
let car = {
    make: "Toyota",
    year: 2023,
    startEngine: function() {
      console.log("The engine has started.");
    }
};

// Deleting the startEngine method
delete car.startEngine;

// Trying to call the deleted method
console.log(car.startEngine);
// Output: undefined
```

That’s it!

Newton School of Technology

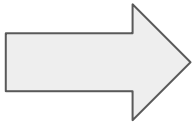
Iterating over an Object

Why Iterate Over Objects?

Newton School of Technology

Objects store key-value pairs, but unlike arrays, they lack built-in iteration methods. We need to loop through an object's properties to access each key and value.

key	value
name	Ajay
age	30
occupation	Engineer

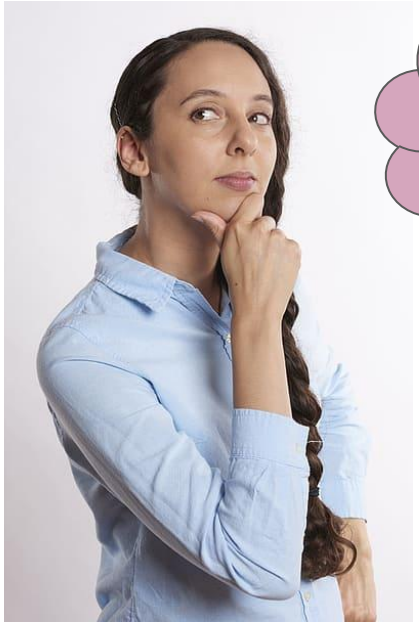


We may need to iterate over this object to display its values, but there are also many practical scenarios where iteration is essential.

How to Iterate: for...in loop

Newton School of Technology

We can iterate over an object using for...in loop



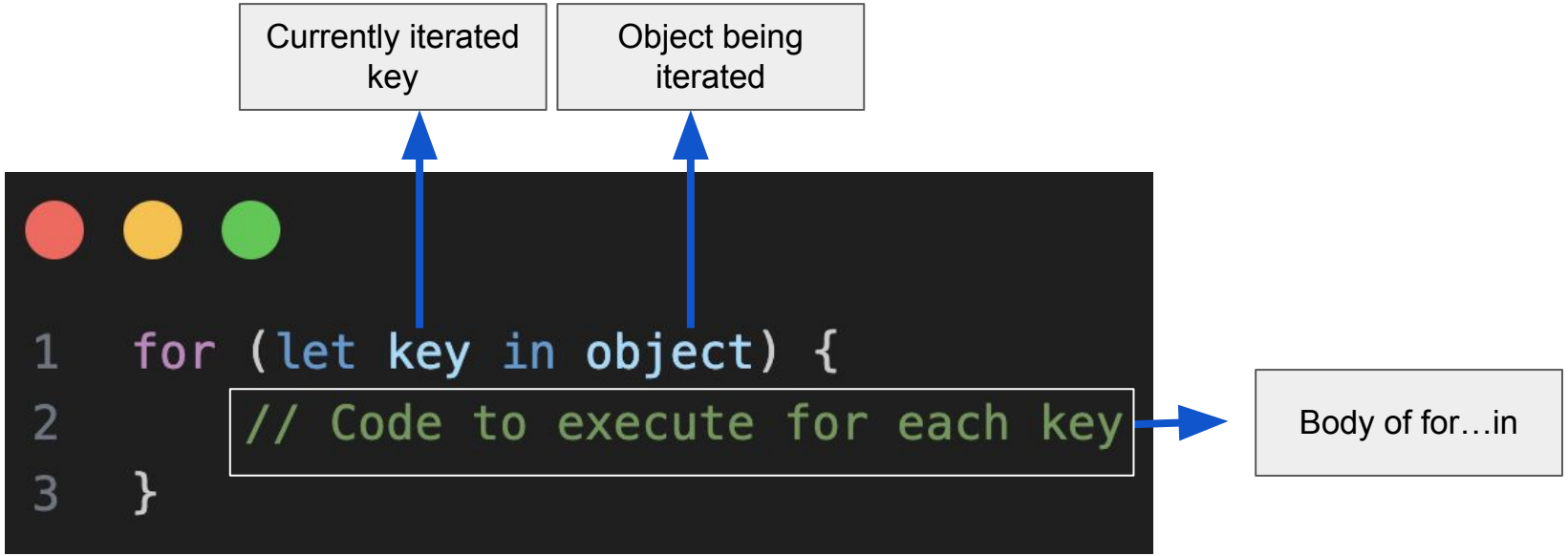
But how to iterate objects??

for...in loop is here for rescue

Iterate Object: for...in loop

Newton School of Technology

The for...in loop is used to iterate over the keys of an object, allowing you to access both the key and its corresponding value.



for...in: Example

Newton School of Technology

Let's iterate over the person object (with name, age, and occupation) and display its keys and values.

```
1  const person = {
2      name: "Ajay",
3      age: 30,
4      occupation: "Engineer"
5  };
6
7  for (let key in person) {
8    console.log(key, ":", person[key]);
9  }
```

Output:

name : Ajay
age: 30
occupation: Engineer

Iterate: for...of loop

Newton School of Technology

Plain objects ({}) are not directly iterable with for...of. However, we can make them iterable by using methods like Object.keys().

```
1  const obj = { a: 1, b: 2, c: 3 };
2
3  // Iterating over the object using Object.keys
4  for (let key of Object.keys(obj)) {
5      console.log(key); // Outputs: a, b, c
6  }
```

Output:

a
b
c

In Class Questions

Newton School of Technology

Thanks for watching!