# JavaScript Objects Quick Start Guide

```
const person = {
  firstName: 'John',
  lastName: 'Doe',
  age: 30,
```

JavaScript objects are a core data structure used to store and organize data. They are collections of key-value pairs, where each key is a unique string (or symbol in modern JavaScript) that maps to a corresponding value. Objects are versatile and commonly used in JavaScript to represent complex data structures. Here's a detailed explanation of how JavaScript objects work, along with examples:

## Creating an Object

You can create an object in JavaScript using object literal notation, which consists of curly braces {} and key-value pairs:

```
const person = {
  firstName: 'John',
  lastName: 'Doe',
  age: 30,
};
```

## Accessing Object Properties

You can access properties of an object using dot notation or bracket notation:

```
console.log(person.firstName); // 'John'
console.log(person['lastName']); // 'Doe'
```

## Setting Object Properties

You can set or update the value of an object property using assignment:

```
person.age = 31;
person['firstName'] = 'Jane';

console.log(person); // { firstName: 'Jane', lastName:
'Doe', age: 31 }
```

## Adding New Properties

You can add new properties to an existing object simply by assigning a value to a new key:

```
person.email = 'jane@example.com';

console.log(person); // { firstName: 'Jane', lastName:
'Doe', age: 31, email: 'jane@example.com' }
```

## Removing Properties

You can remove properties from an object using the delete keyword:

```
delete person.age;

console.log(person); // { firstName: 'Jane', lastName:
'Doe', email: 'jane@example.com' }
```

## Iterating Over Object Properties

You can loop through an object's properties using for...in loops or methods like Object.keys(), Object.values(), and Object.entries():

```
for (const key in person) {
  console.log(key, person[key]);
}

const keys = Object.keys(person);
console.log(keys); // ['firstName', 'lastName',
'email']

const values = Object.values(person);
```

```
console.log(values); // ['Jane', 'Doe',
'jane@example.com']

const entries = Object.entries(person);
console.log(entries);
// [['firstName', 'Jane'], ['lastName', 'Doe'],
['email', 'jane@example.com']]
```

## Objects with Methods

Objects can also contain functions as properties, which are known as methods:

```
const calculator = {
  add: function (x, y) {
    return x + y;
  },
  subtract: function (x, y) {
    return x - y;
  },
};

console.log(calculator.add(5, 3)); // 8
console.log(calculator.subtract(5, 3)); // 2
```

## Objects as Containers

Objects can serve as containers for related data and behavior, making them a powerful tool for organizing and managing data in JavaScript. They're commonly used to represent things like user profiles, products, configurations, and more.
In summary, JavaScript objects are essential data structures that use key-value pairs to organize and store data. Understanding how to create, access, and manipulate objects is crucial for building complex applications in JavaScript.
These exercises will help reinforce your understanding of JavaScript objects and their manipulation. Feel free to experiment with more complex objects and scenarios as you become more comfortable with these concepts.

## Example 1: Creating and Accessing Object Properties

```
// Creating an object
const person = {
  firstName: 'John',
  lastName: 'Doe',
  age: 30,
};

// Accessing object properties
```

```
console.log(person.firstName); // Exercise: Access and
print the last name
console.log(person.age);      // Exercise: Access and
print the age
```

Exercise 1: Access and print the last name and age of the person object.

## Example 2: Setting and Modifying Object Properties

```
const car = {
  make: 'Toyota',
  model: 'Camry',
};

// Setting a new property
car.year = 2022;

// Modifying an existing property
car.model = 'Corolla';

console.log(car); // Exercise: Add a property 'color'
with the value 'blue' and modify 'make' to 'Honda'
```

Exercise 2: Add a property named 'color' with the value 'blue' to the car object. Then, modify the 'make' property to 'Honda'.

## Example 3: Object with Methods

```
const rectangle = {
  width: 5,
  height: 10,
  // Method to calculate area
  calculateArea: function () {
    return this.width * this.height;
  },
};

console.log(rectangle.calculateArea()); // Exercise:
Calculate and print the rectangle's perimeter
```

Exercise 3: Add a method to the rectangle object that calculates and returns the rectangle's perimeter (2 * width + 2 * height). Call the method and print the perimeter.

## Example 4: Iterating Over Object Properties

```
const student = {
  name: 'Alice',
  age: 25,
  grade: 'A',
};

// Iterate over object properties
for (const key in student) {
  console.log(`${key}: ${student[key]}`);
}

// Exercise: Create a function that prints all
properties and values of an object
```

Exercise 4: Create a function called printObject that takes an object as a parameter and prints all its properties and values. Then, use this function to print the properties and values of the student object.

## Example 5: Deleting Object Properties

```
const computer = {
  brand: 'Dell',
  model: 'XPS 13',
  year: 2021,
};

// Deleting a property
delete computer.year;

console.log(computer); // Exercise: Delete the 'model'
property
```

Exercise 5: Delete the 'model' property from the computer object using the delete keyword.

## Example 6: Nested Objects

```
const address = {
  street: '123 Main St',
  city: 'Anytown',
  zipCode: '12345',
};
```

```
const person = {
  firstName: 'Alice',
  lastName: 'Johnson',
  age: 28,
  contact: address,
};

// Exercise: Access and print the city of the person's
address
```

Exercise 6: Access and print the city of the person object's address.

## Example 7: Object Methods and "this"

```
const bankAccount = {
  balance: 1000,
  deposit: function (amount) {
    this.balance += amount;
  },
  withdraw: function (amount) {
    this.balance -= amount;
  },
};

bankAccount.deposit(500);
bankAccount.withdraw(200);

console.log(bankAccount.balance); // Exercise: Add a
method to check the account balance
```

Exercise 7: Add a method called checkBalance to the bankAccount object that returns the current balance. Call the method and print the balance.

## Example 8: Object Constructors

```
function Dog(name, breed, age) {
  this.name = name;
  this.breed = breed;
  this.age = age;
}

const dog1 = new Dog('Buddy', 'Golden Retriever', 3);
const dog2 = new Dog('Max', 'German Shepherd', 2);
```

```
console.log(dog1); // Exercise: Create a third dog
object and print its details
```

Exercise 8: Create a third dog3 object using the Dog constructor and print its details.

## Example 9: Object Destructuring

```
const student = {
  firstName: 'Emily',
  lastName: 'Smith',
  age: 21,
};

const { firstName, lastName } = student;

console.log(`${firstName} ${lastName}`); // Exercise:
Create an object with more properties and use
destructuring to extract them
```

Exercise 9: Create an object with additional properties (e.g., email, major) and use object destructuring to extract and print those properties.

## Example 10: Object Equality

```
const person1 = {
  name: 'Alice',
  age: 30,
};

const person2 = {
  name: 'Alice',
  age: 30,
};

console.log(person1 === person2); // Exercise: Compare
two objects for equality
```

Exercise 10: Compare two objects for equality. Create two more objects with different property values and compare them for equality as well.

*These exercises should further enhance your understanding of JavaScript objects and how to work with them.*