ScopeTreasure Courses

JAVASCRIPT COURSES · LESSON 3.9 · CODE SCOPE SERIES

Objects

Data with named boxes: properties, dot vs bracket, changing and deleting, methods and this, objects inside arrays, and for...in.

Introduction

One cup of chai has a name, a price, and a stock status. With plain variables you scatter them: `chaiName`, `chaiPrice`, `chaiStock` — three loose boxes that clearly belong together, and chaos when a second chai arrives. An **object** gathers them into **one bundle with named boxes**.

In this lesson you will build an object, read it with dot and bracket, change values, add and delete properties, give the object its own function — a **method** — and meet the shape of almost all real app data: **objects inside an array**.

(Roman Urdu: Object matlab naam walay khaanon ka aik bundle — aik cheez ki saari maloomat aik jagah.)

Key points

- **An object is a bundle of named boxes:** `let chai = { name: "Doodh Patti", price: 250, inStock: true };` — each box has a **key** (the name) and a **value**, joined by `:`, separated by commas, inside curly brackets. Each key-value pair is called a **property**.
- **Read with dot notation:** `chai.price` reads the `price` box — `250`. Dot is short and clear, and it is what you will use most of the time.

  (Roman Urdu: Dot lagao, key ka naam likho — value mil jayegi, jaise chai.price.)
- **Bracket notation does the same job:** `chai["price"]` is also `250`. Brackets are **needed when the key is stored in a variable**: `let key = "price"; chai[key];` — dot cannot do this, because `chai.key` would look for a box literally named "key".
- **Change, add, delete — objects are flexible:** `chai.price = 300;` changes a value. `chai.size = "Large";` adds a **brand-new property** after creation. `delete chai.size;` removes one.
- **A method is a function inside the object:** when a property's value is a function, it is called a **method**. Inside a method, **`this` means THIS object** — `this.price` reads the object's own price, whichever object the method belongs to.

  (Roman Urdu: Method ke andar "this" ka matlab hai — yehi wala dabba, is ki apni values.)
- **Arrays of objects = real app data:** a menu is an array of objects — `menu[1].price` walks into the second object and reads its price. First the array index, then the object's key.
- **`for...in` walks every key:** `for (let key in chai)` gives you each key name one by one, so you can read or print every property without knowing them in advance.
- **Array vs object:** an array is an **ordered list found by number**; an object is a **named bundle found by key**. Lists → array. One thing with named details → object.

Code examples

Example 1 — Build an object, read it both ways

```
let chai = {
  name: "Doodh Patti",
  price: 250,
  inStock: true
};

console.log(chai.name);
console.log(chai.price);
console.log(chai["inStock"]);

let key = "price";
console.log(chai[key]);
```

What it does: One object holds the chai's three facts together. The first two reads use dot notation; the third uses brackets with a written key; the last uses brackets with the key stored in a variable — the job only brackets can do.

What you will see: four lines in the Console: `Doodh Patti`, `250`, `true`, and `250`.

Example 2 — Change, add, delete, and a method with this

```
let chai = {
  name: "Doodh Patti",
  price: 250,
  inStock: true,
  describe: function () {
    return this.name + " costs " + this.price + " rupees";
  }
};

chai.price = 300;
chai.size = "Large";
delete chai.inStock;

console.log(chai.describe());
console.log(chai.size);
console.log(chai.inStock);
```

What it does: The price changes to `300`, a new property `size` is added after creation, and `inStock` is deleted. The method `describe` builds a sentence using `this` — the object's own name and its **new** price.

What you will see: three lines in the Console: `Doodh Patti costs 300 rupees`, then `Large`, then `undefined` — the deleted property is truly gone.

Example 3 — A menu of objects, and for...in

```
let menu = [
  { name: "Doodh Patti", price: 250 },
  { name: "Cold Coffee", price: 200 },
  { name: "Sweet Lassi", price: 180 }
];

console.log(menu[1].name);
console.log(menu[1].price);

let chai = { name: "Doodh Patti", price: 250, inStock: true };

for (let key in chai) {
  console.log(key + ": " + chai[key]);
}
```

What it does: Three objects live inside one array — the shape of real app data. `menu[1]` picks the second object, and `.price` reads inside it. Then `for...in` walks every key of the single `chai` object and prints each key with its value, using bracket notation because the key arrives in a variable.

What you will see: five lines in the Console: `Cold Coffee`, `200`, then `name: Doodh Patti`, `price: 250`, and `inStock: true`.

Quick recap

- An object is a **named bundle**: keys and values inside curly brackets, read with `chai.price` or `chai["price"]`.
- Properties can be changed, added, and deleted after creation; a **method** is a function inside, and `this` means this object.
- Arrays of objects are the shape of real data — `menu[1].price` — and `for...in` walks every key.

Common mistakes

1. **Using a variable key with dot notation** — `chai.key` looks for a property literally named "key" and gives `undefined`. Fix: when the key is in a variable, use brackets: `chai[key]`.
2. **Forgetting `this` inside a method** — writing just `price` inside the method does not find the object's price and throws a `ReferenceError`. Fix: inside a method, always reach the object's own values with `this.price`, `this.name`, and so on.
3. **Using an array where an object belongs** — storing one item's details by position (`item[0]`, `item[1]`) is hard to read and easy to mix up. Fix: one thing with named details = object; a list of things = array —

and combine them as an array of objects.

Practice

1. Make an object for a book with `title`, `author`, and `pages` properties. Print it with dot notation, then change the pages, add a new property `language`, and print the whole object again.
2. Make an array of three student objects, each with `name` and `marks`. Use a loop to print each student's name and marks, and add a method `result` to one object that uses `this.marks` to return `"Pass"` if marks are `40` or more, else `"Fail"`.

Practice quiz

1. For let chai = { name: "Doodh Patti", price: 250, inStock: true };, what is chai.price?

- "Doodh Patti"
- 250
- true
- undefined

Answer: 250

2. The key you want is stored in a variable: let key = "price";. How do you read it?

- chai.key, which reads the variable
- chai[key], because dot notation looks for a property literally named key
- chai.(key)
- key.chai

Answer: chai[key], because dot notation looks for a property literally named key

3. Inside an object method, what does this mean?

- The global window object, always
- This object itself, so this.price reads the object's own price
- The function's parameter
- Nothing, this is not allowed in methods

Answer: This object itself, so this.price reads the object's own price

---

Watch the video lesson: https://youtu.be/dpzIh_iZ8II

Read online: https://scopetreasure.com/notes/js-objects · More free tools and notes at https://scopetreasure.com