**✅ JavaScript Complete Topics List (for Notes)**

**1. Introduction to JavaScript**

* What is JavaScript?
* Uses of JavaScript
* How to run JavaScript in HTML
* Console.log, alert, prompt, confirm

**2. JavaScript Basics**

* Variables (var, let, const)
* Data Types
* Operators (Arithmetic, Assignment, Comparison, Logical)
* Comments
* Type Conversion (Implicit & Explicit)

**3. Control Flow**

* if, else if, else
* switch statement
* Ternary Operator

**4. Loops**

* for loop
* while loop
* do-while loop
* break & continue

**5. Functions**

* Function declaration
* Function expression
* Parameters vs Arguments
* Return value
* Arrow functions

**6. Objects**

* Object creation
* Accessing properties
* Adding/deleting properties
* Nested objects

**7. Arrays**

* Array creation
* Accessing and modifying elements
* Array methods (push, pop, shift, unshift, splice, slice, etc.)

**8. String Handling**

* String methods (length, toUpperCase, indexOf, includes, etc.)
* Template Literals

**9. Scope and Hoisting**

* Global vs Local Scope
* Hoisting of variables and functions

**10. The this keyword**

* How this works in different contexts

**11. DOM Manipulation**

* Selecting elements (getElementById, querySelector, etc.)
* Changing content and style
* Adding and removing elements

**12. Events**

* Event types (click, input, change, submit)
* addEventListener
* Event object

**13. Form Handling**

* Capturing form input
* Preventing default behavior
* Basic validation

**14. Error Handling**

* try-catch block
* throw keyword

**15. ES6+ Features**

* let and const
* Template strings
* Destructuring
* Spread and Rest operator
* Default parameters
* Modules (import/export)

**16. JSON & Fetch API**

* What is JSON
* Fetching data from API
* Converting JSON

**17. LocalStorage & SessionStorage**

**18. Advanced Topics (Optional)**

* Closures
* Callbacks
* Promises
* Async/Await

**🧠 JavaScript Scope (স্কোপ)**

**🔹 Scope মানে কী?**

**Scope** হলো সেই এলাকা বা জায়গা যেখান থেকে তুমি কোনো ভেরিয়েবল, ফাংশন, বা অবজেক্টকে access (ব্যবহার) করতে পারো।

অর্থাৎ, **কোথা থেকে কোন ভেরিয়েবলকে পাওয়া যাবে আর কোথা থেকে যাবে না**, সেটা নির্ধারণ করে scope।

**🔹 ৩ ধরণের Scope আছে:**

| **Scope Type** | **ব্যাখ্যা** |
| --- | --- |
| 1. **Global Scope** | স্ক্রিপ্টের বাইরে বা উপরে যা declare করা হয় |
| 2. **Function Scope** | ফাংশনের ভিতরে declare করলে সেটা শুধু ঐ ফাংশনের ভিতরেই পাওয়া যায় |
| 3. **Block Scope** | {} এর ভিতর (যেমন if, for, while, etc.) declare করা ভেরিয়েবল |

**🔸 1. Global Scope (সার্বজনীন এলাকা)**

যেকোনো জায়গা থেকে access করা যায়।

javascript

CopyEdit

let country = "Bangladesh";

function showCountry() {

console.log(country); // Access possible

}

console.log(country); // Access possible

**🔸 2. Function Scope (ফাংশনের ভিতর সীমাবদ্ধ)**

var, let, const — সব ফাংশনের ভিতরে declare করলে বাইরে থেকে access করা যায় না।

javascript

CopyEdit

function greet() {

let name = "Rafia";

console.log(name); // OK

}

greet();

console.log(name); // ❌ Error: name is not defined

**🔸 3. Block Scope (Block {} এর ভিতরে সীমাবদ্ধ)**

let ও const block scoped, কিন্তু var না।

javascript

CopyEdit

if (true) {

let x = 10;

const y = 20;

var z = 30;

}

console.log(z); // ✅ var works outside block

console.log(x); // ❌ Error: let is block scoped

console.log(y); // ❌ Error: const is block scoped

**📌 Scope Difference Chart**

| **Keyword** | **Function Scope** | **Block Scope** |
| --- | --- | --- |
| var | ✅ | ❌ |
| let | ✅ | ✅ |
| const | ✅ | ✅ |

**🧪 Practical Example:**

javascript

CopyEdit

let a = 5;

function test() {

let b = 10;

if (true) {

let c = 15;

console.log(a); // 5 (global)

console.log(b); // 10 (function)

console.log(c); // 15 (block)

}

// console.log(c); // ❌ Error

}

test();

**📝 Summary:**

* 🔵 **Global Scope** – সব জায়গা থেকে দেখা যায়
* 🟢 **Function Scope** – শুধুমাত্র ফাংশনের ভিতরে
* 🟡 **Block Scope** – if, for, while ইত্যাদির {} এর ভিতরে সীমাবদ্ধ

**✅ Bonus Tip:**

যখনই সম্ভব **let বা const ব্যবহার করো**, কারণ এগুলো predictable এবং safer।

**🧠 JavaScript Expressions (এক্সপ্রেশন)**

**🔹 Expression মানে কী?**

**Expression** হলো এমন কোনো জাভাস্ক্রিপ্ট কোডের অংশ যা **একটি মান (value)** রিটার্ন করে।

সহজ ভাষায়: Expression মানে এমন কিছু কোড যা “evaluate” করে একটি value দেয়।

**✅ উদাহরণ ১: সাধারণ Expression**

javascript

CopyEdit

5 + 10 // এটা একটি expression → result হবে 15

"a" + "b" // এটা একটি expression → result হবে "ab"

true || false // result হবে true

**🔍 Expression-এর ধরন**

JavaScript-এ অনেক রকম expression আছে। নিচে প্রতিটি ব্যাখ্যা সহ দেওয়া হলো:

**1. Arithmetic Expression (গাণিতিক এক্সপ্রেশন)**

গাণিতিক কাজ করে এবং রেজাল্ট দেয়:

javascript

CopyEdit

let result = 10 + 5; // 15

**2. String Expression (স্ট্রিং এক্সপ্রেশন)**

javascript

CopyEdit

let fullName = "Rafia" + " Akter"; // "Rafia Akter"

**3. Logical Expression (লজিক্যাল এক্সপ্রেশন)**

javascript

CopyEdit

true && false // false

true || false // true

**4. Assignment Expression**

javascript

CopyEdit

let x = 10;

👉 এখানে x = 10 হলো একটা expression কারণ এটা একটি মান return করে (value: 10) এবং x তে সেট করে।

**5. Comparison Expression (তুলনামূলক)**

javascript

CopyEdit

5 > 3 // true

10 == 20 // false

**6. Function Expression**

ফাংশনকে একটি ভেরিয়েবলে রেখে দেওয়া যায় — এটাকে বলে Function Expression:

javascript

CopyEdit

const greet = function() {

return "Hello";

}

👉 এখানে পুরো function() { return "Hello"; } এটা একটা expression, কারণ এটি একটি মান হিসেবে ব্যবহার হয়েছে।

**🆚 Expression vs Statement**

| **বিষয়** | **Expression** | **Statement** |
| --- | --- | --- |
| সংজ্ঞা | মান দেয় | কাজ করে |
| উদাহরণ | 5 + 10, "a" + "b" | if, for, function, etc. |
| শেষ হয় | value দিয়ে | semicolon দিয়ে |

✅ Example:

javascript

CopyEdit

let x = 5 + 10; // পুরোটা একটা statement, কিন্তু 5 + 10 = expression

**🧪 Practice:**

javascript

CopyEdit

let a = 2 + 3; // Expression → evaluates to 5

let b = a > 4; // Expression → evaluates to true

let c = function() {}; // Function Expression

**📌 Summary:**

* Expression = এমন কোড যা কোনো **value** রিটার্ন করে।
* এটি যেকোনো জায়গায় ব্যবহার করা যায় যেখানে কোনো মানের প্রয়োজন হয়।
* বিভিন্ন ধরনের expression আছে — arithmetic, logical, assignment, string, function, etc.
* **🧠 JavaScript Operators (অপারেটরস)**
* **🔹 অপারেটর কী?**
* **Operator** হলো একটি চিহ্ন (symbol) বা শব্দ (keyword) যা ভেরিয়েবল ও মানের উপর কাজ করে এবং একটি নতুন মান রিটার্ন করে।
* উদাহরণ:
* javascript
* CopyEdit
* let result = 10 + 5;
* এখানে + হল operator, যা ১০ আর ৫-কে যোগ করে ১৫ রিটার্ন করছে।
* **🧮 JavaScript-এ বিভিন্ন রকমের Operator আছে:**

| **Operator Type** | **ব্যাখ্যা** |
| --- | --- |
| 1. Arithmetic Operators | গাণিতিক কাজ |
| 2. Assignment Operators | মান সেট করা |
| 3. Comparison Operators | তুলনা করা |
| 4. Logical Operators | সত্য/মিথ্যার ভিত্তিতে কাজ |
| 5. Unary Operators | একটিমাত্র মানের উপর কাজ করে |
| 6. Ternary Operator | শর্টকাট if-else |
| 7. String Concatenation | স্ট্রিং জোড়া দেওয়া |
| 8. Bitwise Operators (Advanced) | বিট লেভেলে কাজ |

* **🔸 1. Arithmetic Operators (গাণিতিক অপারেটর)**

| **Operator** | **কাজ** | **উদাহরণ** |
| --- | --- | --- |
| + | যোগ | 5 + 3 = 8 |
| - | বিয়োগ | 5 - 2 = 3 |
| \* | গুণ | 4 \* 2 = 8 |
| / | ভাগ | 10 / 2 = 5 |
| % | ভাগশেষ | 10 % 3 = 1 |
| \*\* | ঘাত | 2 \*\* 3 = 8 |
| ++ | ইনক্রিমেন্ট | x++ মান ১ বাড়ায় |
| -- | ডিক্রিমেন্ট | x-- মান ১ কমায় |

* **🔸 2. Assignment Operators (মান সেট করা)**

| **Operator** | **কাজ** | **উদাহরণ** |
| --- | --- | --- |
| = | সেট করা | x = 10 |
| += | যোগ করে সেট | x += 5 → x = x + 5 |
| -= | বিয়োগ করে সেট | x -= 2 |
| \*= | গুণ করে সেট | x \*= 3 |
| /= | ভাগ করে সেট | x /= 2 |
| %= | ভাগশেষ করে সেট | x %= 3 |

* **🔸 3. Comparison Operators (তুলনামূলক অপারেটর)**

| **Operator** | **ব্যাখ্যা** | **উদাহরণ** | **রিটার্ন** |
| --- | --- | --- | --- |
| == | সমান কি না (type না দেখে) | 5 == '5' | ✅ true |
| === | একেবারে সমান (type সহ) | 5 === '5' | ❌ false |
| != | সমান না | 5 != 3 | true |
| !== | type সহ সমান না | 5 !== '5' | true |
| > | বড় কি না | 5 > 3 | true |
| < | ছোট কি না | 2 < 4 | true |
| >= | বড় বা সমান | 5 >= 5 | true |
| <= | ছোট বা সমান | 3 <= 2 | false |

* **🔸 4. Logical Operators (যুক্তি নির্ভর অপারেটর)**

| **Operator** | **নাম** | **কাজ** | **উদাহরণ** |
| --- | --- | --- | --- |
| && | AND | দুইটা সত্য হলে সত্য | true && true → true |
| ` |  | ` | OR |
| ! | NOT | সত্য কে মিথ্যা বানায় | !true → false |

* **🔸 5. Unary Operator**
* একটি মাত্র মানের উপর কাজ করে।

| **Operator** | **কাজ** | **উদাহরণ** |
| --- | --- | --- |
| typeof | ভেরিয়েবলের টাইপ দেখায় | typeof 5 → "number" |
| ++, -- | পূর্বে দেখা হয়েছে |  |

* **🔸 6. Ternary Operator (Shortcut if-else)**
* javascript
* CopyEdit
* let age = 18;
* let result = (age >= 18) ? "Adult" : "Minor";
* console.log(result); // Adult
* 👉 Syntax:
* javascript
* CopyEdit
* (condition) ? trueResult : falseResult;
* **🔸 7. String Concatenation Operator**
* javascript
* CopyEdit
* let firstName = "Rafia";
* let lastName = "Akter";
* let fullName = firstName + " " + lastName;
* console.log(fullName); // Rafia Akter
* **📌 Summary Table**

| **টাইপ** | **অপারেটর উদাহরণ** | **কাজ** |
| --- | --- | --- |
| Arithmetic | + - \* / % ++ | সংখ্যা নিয়ে কাজ |
| Assignment | = += -= | মান সেট করা |
| Comparison | == != === > < | তুলনা করা |
| Logical | `&& |  |
| Ternary | condition ? x : y | শর্টকাট if-else |
| Unary | typeof | একক মানের উপর কাজ |
| String | + | স্ট্রিং জোড়া দেওয়া |

* **🧪 Practice Task:**
* javascript
* CopyEdit
* let a = 10;
* let b = 5;
* console.log(a + b); // 15
* console.log(a > b); // true
* console.log(a == '10'); // true
* console.log(typeof a); // number
* console.log(a >= 10 ? "OK" : "Not OK"); // OK

**🧠 JavaScript Statements (স্টেটমেন্ট)**

**🔹 Statement মানে কী?**

**JavaScript Statement** হলো এক বা একাধিক JavaScript **instruction বা কাজের নির্দেশনা** যা browser বা interpreter একে একে (top to bottom) execute করে।

সহজ ভাষায়:  
👉 একেকটা statement মানে JavaScript কে “কি করতে হবে” সেই নির্দেশ দেওয়া।

**✅ Example:**

javascript

CopyEdit

let a = 10; // এটা একটা statement

console.log(a); // এটাও একটা statement

প্রতিটি statement শেষ হয় semicolon (;) দিয়ে (optional হলেও Best Practice হল ব্যবহার করা)।

**📚 Statement vs Expression**

| **বিষয়** | **Expression** | **Statement** |
| --- | --- | --- |
| সংজ্ঞা | একটি মান রিটার্ন করে | কোনো কাজ বা instruction দেয় |
| শেষ হয় | একটি মানে | সেমিকোলনে |
| উদাহরণ | 5 + 3, "Hello" + " World" | let x = 10;, if(...), for(...) |

**🔸 JavaScript-এর ধরণের Statements**

**1. Declaration Statements**

ভেরিয়েবল, কনস্ট্যান্ট ইত্যাদি ঘোষণা করা।

javascript

CopyEdit

let name = "Rafia";

const pi = 3.14;

**2. Conditional Statements (শর্তভিত্তিক)**

নির্দিষ্ট শর্ত অনুযায়ী code block চালায়।

javascript

CopyEdit

if (age >= 18) {

console.log("Adult");

} else {

console.log("Minor");

}

🧾 Includes:

* if
* if...else
* else if
* switch

**3. Looping Statements (পুনরাবৃত্তি)**

কোড বারবার চালানোর জন্য।

javascript

CopyEdit

for (let i = 0; i < 5; i++) {

console.log(i);

}

🧾 Includes:

* for
* while
* do...while
* break, continue

**4. Function Statements**

ফাংশন ঘোষণা ও এক্সিকিউশন

javascript

CopyEdit

function greet() {

console.log("Hello!");

}

greet(); // calling

**5. Try-Catch (Error Handling Statements)**

ভুল ধরা ও হ্যান্ডেল করা।

javascript

CopyEdit

try {

let x = y + 1; // Error

} catch (error) {

console.log("Error found:", error.message);

}

**6. Return Statement**

ফাংশনের ভিতরে ব্যবহার করে কোনো মান ফেরত দিতে।

javascript

CopyEdit

function sum(a, b) {

return a + b;

}

**🔎 Statement Structure**

প্রত্যেকটি স্টেটমেন্ট সাধারণত ৩টি অংশে হয়:

javascript

CopyEdit

let x = 5;

1. **Keyword** → let
2. **Variable Name** → x
3. **Value or Expression** → 5

**📌 Statement Writing Best Practices**

✅ প্রতিটি statement শেষে ; ব্যবহার করো  
✅ লজিক স্পষ্ট করে লেখো  
✅ Nested statement হলে {} ব্যবহার করো

**🧪 Practice:**

javascript

CopyEdit

let marks = 80;

if (marks >= 40) {

console.log("Pass");

} else {

console.log("Fail");

}

**📋 Summary:**

| **ধরণ** | **ব্যাখ্যা** | **উদাহরণ** |
| --- | --- | --- |
| Declaration | Variable বা function ঘোষনা | let x = 5; |
| Conditional | শর্ত দিয়ে code চালানো | if, switch |
| Looping | পুনরাবৃত্তি | for, while |
| Function | ফাংশন চালানো | function greet() |
| Error Handling | ভুল ধরা | try...catch |
| Return | মান ফেরত দেওয়া | return a + b; |

**🧠 JavaScript Object (অবজেক্ট)**

**🔹 Object কী?**

**Object** হলো একটি collection বা সমষ্টি — যেখানে **key:value** জোড়ার মাধ্যমে ডেটা সংরক্ষণ করা হয়।

সহজ ভাষায়: Object হচ্ছে এমন একটা "box", যার মধ্যে তুমি নাম দিয়ে যেকোনো রকমের তথ্য রাখতে পারো।

**✅ Basic Example:**

javascript

CopyEdit

let person = {

name: "Rafia",

age: 22,

isStudent: true

};

এখানে person হচ্ছে object, যার ভিতরে তিনটা property আছে:

* name → "Rafia"
* age → 22
* isStudent → true

**🔸 Object Syntax**

javascript

CopyEdit

let objectName = {

key1: value1,

key2: value2,

key3: value3

};

**🔍 Property Access করার ২টা উপায়:**

**1. Dot Notation (সোজা ও বেশি ব্যবহারযোগ্য)**

javascript

CopyEdit

console.log(person.name); // "Rafia"

**2. Bracket Notation (key string আকারে দিলে)**

javascript

CopyEdit

console.log(person["age"]); // 22

👉 যখন key ডাইনামিক বা স্পেস থাকে, তখন bracket notation দরকার হয়।

**🛠️ Object-এর Property Change/Add/Delete**

javascript

CopyEdit

// Change

person.age = 23;

// Add

person.gender = "Female";

// Delete

delete person.isStudent;

**🔁 Loop Through Object (for...in)**

javascript

CopyEdit

for (let key in person) {

console.log(key + ":", person[key]);

}

**🔸 Nested Object (অবজেক্টের ভিতরে অবজেক্ট)**

javascript

CopyEdit

let student = {

name: "Rafia",

marks: {

math: 90,

science: 85

}

};

console.log(student.marks.math); // 90

**🔸 Object in Array (অ্যারে এর ভিতরে অবজেক্ট)**

javascript

CopyEdit

let users = [

{ name: "Rafia", age: 22 },

{ name: "Ayesha", age: 20 }

];

console.log(users[0].name); // "Rafia"

**🔸 Functions Inside Object → Method**

javascript

CopyEdit

let user = {

name: "Rafia",

greet: function() {

return "Hello, " + this.name;

}

};

console.log(user.greet()); // Hello, Rafia

🔎 this.name এখানে user object-এর name property কে বোঝায়।

**📋 Object Summary Table:**

| **কাজ** | **কোড** |
| --- | --- |
| Create Object | let obj = { key: value }; |
| Access | obj.key বা obj["key"] |
| Add/Update | obj.newKey = value; |
| Delete | delete obj.key; |
| Loop | for (let key in obj) |
| Function Inside | obj.method = function() {} |

**🧪 Practice Task:**

javascript

CopyEdit

let car = {

brand: "Toyota",

model: "Corolla",

year: 2022

};

car.color = "White";

car.year = 2023;

delete car.model;

console.log(car);

**📌 মনে রাখো:**

* Object হলো key-value pair এর ডেটা স্টোরের পদ্ধতি।
* Dot এবং Bracket notation দুটোই জানা জরুরি।
* তুমি অবজেক্টের মধ্যে ফাংশনও রাখতে পারো।

**🧠 JavaScript Array (অ্যারে)**

**🔹 Array কী?**

JavaScript এ **Array** হলো একটি বিশেষ ধরণের অবজেক্ট যা **একাধিক মান (value)** একই ভেরিয়েবলে **index আকারে** সংরক্ষণ করে।

সহজভাবে বললে:  
👉 Array হলো অনেকগুলো মানের “list” বা “collection”, যেগুলো একটি ভেরিয়েবলে রাখা হয়।

**✅ Basic Example:**

javascript

CopyEdit

let fruits = ["Apple", "Banana", "Mango"];

এখানে fruits অ্যারেটির মধ্যে ৩টি মান আছে:

* Index 0 → "Apple"
* Index 1 → "Banana"
* Index 2 → "Mango"

**🔸 Array Declaration & Initialization**

**1. Literal Notation (সবচেয়ে সহজ ও ব্যবহৃত)**

javascript

CopyEdit

let numbers = [10, 20, 30, 40];

**2. Using new Array() Constructor**

javascript

CopyEdit

let names = new Array("Rafia", "Ayesha", "Sadia");

👉 কিন্তু সাধারণত literal [ ] ব্যবহার করাই ভালো।

**🔹 Accessing Array Elements**

javascript

CopyEdit

let colors = ["Red", "Green", "Blue"];

console.log(colors[0]); // Red

console.log(colors[2]); // Blue

📝 Index **0 থেকে শুরু হয়**।  
অ্যারে লেন্থ colors.length = 3

**🔹 Changing & Adding Elements**

javascript

CopyEdit

colors[1] = "Yellow"; // Green → Yellow

colors[3] = "Purple"; // নতুন item যুক্ত হল

**🔹 Array Length**

javascript

CopyEdit

console.log(colors.length); // মোট কতটি item আছে

**🔁 Array Iteration Techniques (Loop দিয়ে ঘোরা)**

**1. for loop:**

javascript

CopyEdit

for (let i = 0; i < fruits.length; i++) {

console.log(fruits[i]);

}

**2. for...of loop:**

javascript

CopyEdit

for (let fruit of fruits) {

console.log(fruit);

}

**3. forEach() method:**

javascript

CopyEdit

fruits.forEach(function(fruit) {

console.log(fruit);

});

**🔧 Common Array Methods (Must Know)**

| **Method** | **কাজ** | **Example** |
| --- | --- | --- |
| push() | শেষে item যোগ | arr.push("Lemon") |
| pop() | শেষ item বাদ | arr.pop() |
| unshift() | শুরুতে item যোগ | arr.unshift("Guava") |
| shift() | শুরু থেকে item বাদ | arr.shift() |
| length | item এর সংখ্যা | arr.length |
| indexOf() | item এর index খুঁজে | arr.indexOf("Mango") |
| includes() | item আছে কিনা | arr.includes("Banana") |
| join() | string বানায় | arr.join(", ") |
| slice(start, end) | অংশ কেটে নেয় (original থাকে) | arr.slice(1, 3) |
| splice(start, deleteCount, newItem) | অংশ কেটে যোগ করে | arr.splice(2, 1, "Kiwi") |
| reverse() | উল্টে দেয় | arr.reverse() |
| sort() | Alphabetically sort | arr.sort() |

**🧪 Example:**

javascript

CopyEdit

let arr = ["A", "B", "C", "D"];

arr.push("E"); // শেষে যোগ → ["A", "B", "C", "D", "E"]

arr.pop(); // শেষ বাদ → ["A", "B", "C", "D"]

arr.shift(); // শুরু বাদ → ["B", "C", "D"]

arr.unshift("Z"); // শুরুতে যোগ → ["Z", "B", "C", "D"]

arr.splice(2, 1, "X"); // index 2 তে "C" বাদ দিয়ে "X" বসাও → ["Z", "B", "X", "D"]

**🔸 Array of Objects (প্রায় ব্যবহার হয়)**

javascript

CopyEdit

let students = [

{ name: "Rafia", age: 22 },

{ name: "Ayesha", age: 20 }

];

console.log(students[0].name); // Rafia

**🔸 Multidimensional Array (Array inside Array)**

javascript

CopyEdit

let matrix = [

[1, 2, 3],

[4, 5, 6]

];

console.log(matrix[1][2]); // 6

**🔍 Check if Variable is Array**

javascript

CopyEdit

Array.isArray(fruits); // true

**🔥 Advanced Methods (Later when ready)**

| **Method** | **কাজ** |
| --- | --- |
| map() | প্রত্যেক item এ function চালিয়ে নতুন array দেয় |
| filter() | শর্ত অনুযায়ী কিছু item বের করে |
| find() | প্রথম matching item দেয় |
| reduce() | সবগুলো item কে একত্র করে একটা মান বানায় |

উদাহরণ:

javascript

CopyEdit

let nums = [1, 2, 3];

let doubled = nums.map(x => x \* 2); // [2, 4, 6]

**📌 Summary Chart**

| **কাজ** | **Syntax** | **ফলাফল** |
| --- | --- | --- |
| Declare | let a = [1, 2] | অ্যারে তৈরি |
| Access | a[0] | প্রথম item |
| Add | a.push(5) | শেষে যোগ |
| Remove | a.pop() | শেষ থেকে বাদ |
| Loop | for, forEach, for...of | ঘুরে দেখা |
| Check | Array.isArray(a) | true/false |

**🧪 Practice Task**

javascript

CopyEdit

let items = ["Pen", "Pencil", "Eraser"];

items.push("Sharpener");

items.splice(1, 1, "Color Pencil");

items.forEach(item => console.log(item));

**📄 Final Thought**

* Array সবসময় index দিয়ে access করতে হয়।
* JavaScript এ Array = Object type.
* অনেক Built-in method থাকায় array নিয়ে কাজ করা খুব সহজ ও শক্তিশালী।

**🧠 JavaScript Function (ফাংশন)**

**🔹 Function কী?**

**Function** হলো এমন একটি কোডের টুকরা (block) যা তুমি একবার লিখে বারবার ব্যবহার করতে পারো।  
তুমি চাইলে সেই ফাংশনকে বিভিন্ন input দিতে পারো, আর সেটা একটি ফলাফল (output) রিটার্ন করতে পারে।

🎯 এক কথায়: Function মানে হলো "Reusable Code Block"

**✅ Basic Function Structure:**

javascript

CopyEdit

function functionName(parameters) {

// code block

return result;

}

**🔸 1. Function Declaration (সবচেয়ে সাধারণ পদ্ধতি)**

javascript

CopyEdit

function greet() {

console.log("Hello, World!");

}

greet(); // Function call

**🔸 2. Function with Parameters**

javascript

CopyEdit

function greetUser(name) {

console.log("Hello, " + name + "!");

}

greetUser("Rafia"); // Output: Hello, Rafia!

👉 এখানে name হলো parameter, আর "Rafia" হলো argument।

**🔸 3. Function with Return**

javascript

CopyEdit

function add(a, b) {

return a + b;

}

let result = add(5, 3);

console.log(result); // 8

📌 return statement ফাংশন থেকে output বের করে আনে।

**🔸 4. Function Expression**

javascript

CopyEdit

const sayHi = function() {

return "Hi!";

};

console.log(sayHi()); // Hi!

👉 এখানে ফাংশনটা একটি ভেরিয়েবলে রাখা হয়েছে।

**🔸 5. Arrow Function (ES6 Feature)**

javascript

CopyEdit

const multiply = (a, b) => {

return a \* b;

};

console.log(multiply(4, 5)); // 20

👉 শর্ট করে লেখার সুবিধা:

javascript

CopyEdit

const square = x => x \* x;

**🔸 6. Default Parameters**

javascript

CopyEdit

function greet(name = "Guest") {

console.log("Hello, " + name);

}

greet(); // Hello, Guest

greet("Rafia"); // Hello, Rafia

**🔸 7. Function Scope (Variable Inside Function)**

javascript

CopyEdit

function showAge() {

let age = 25;

console.log(age);

}

showAge(); // 25

// console.log(age); // ❌ Error: age is not defined

👉 ফাংশনের ভিতরে declare করা ভেরিয়েবল বাইরে থেকে access করা যায় না।

**🔸 8. Nested Functions (ফাংশনের ভিতরে ফাংশন)**

javascript

CopyEdit

function outer() {

function inner() {

console.log("I'm inner");

}

inner();

}

outer(); // I'm inner

**🔸 9. Callback Function**

javascript

CopyEdit

function process(callback) {

callback();

}

process(function() {

console.log("Callback function executed!");

});

👉 একটি ফাংশনকে আরেক ফাংশনের ভিতরে পাঠানো হয় — এটা Web Development এ খুব ব্যবহার হয়।

**🔸 10. Function in Object (Method)**

javascript

CopyEdit

let person = {

name: "Rafia",

greet: function() {

return "Hi, " + this.name;

}

};

console.log(person.greet()); // Hi, Rafia

👉 যখন কোনো ফাংশন object-এর property হয়, তখন তাকে **method** বলে।

**📝 Summary Table:**

| **Function Type** | **Syntax Example** |
| --- | --- |
| Declaration | function fn() {} |
| Expression | const fn = function() {} |
| Arrow Function | const fn = () => {} |
| With Parameters | function greet(name) {} |
| With Return | return result; |
| Default Parameters | function fn(name = "Guest") |
| Callback Function | fn(function() {}) |
| Object Method | obj.fn = function() {} |

**🧪 Practice Task:**

javascript

CopyEdit

function calculator(a, b, operator) {

if (operator === "+") return a + b;

else if (operator === "-") return a - b;

else return "Invalid Operator";

}

console.log(calculator(5, 3, "+")); // 8

console.log(calculator(10, 4, "-")); // 6

**⚠️ Function VS Method**

| **Function** | **Method** |
| --- | --- |
| আলাদা ভাবে ডাকা যায় | অবজেক্টের অংশ হিসেবে |
| function greet() {} | person.greet = function() {} |

**📄 Final Thoughts:**

* Function = কোডের block যা বারবার ব্যবহার করা যায়
* Parameters দিয়ে input, return দিয়ে output
* Arrow function, Callback, Method – এসব Modern JavaScript এ খুবই জরুরি
* যত বেশি ফাংশন ব্যবহার করবে, তত clean এবং reusable হবে তোমার কোড

**১. Class কি?**

JavaScript-এ **Class** হল একটা **টেম্পলেট** বা খসড়া যার মাধ্যমে আমরা একই ধরনের অনেক অবজেক্ট তৈরি করতে পারি।

Class ব্যবহার করে আমরা সহজে অবজেক্টের প্রোপার্টি (property) এবং মেথড (method) ডিফাইন করতে পারি।

ES6 (2015) থেকে JavaScript-এ Class এ নতুন সিনট্যাক্স এসেছে, যা আগের Constructor Function আর Prototype বেইজড কাজের তুলনায় অনেক সহজ ও পরিষ্কার।

**২. Class এর সাধারণ সিনট্যাক্স**

javascript

CopyEdit

class ClassName {

constructor(parameters) {

// প্রোপার্টি initialize করার জন্য কোড

}

methodName() {

// মেথডের কাজ

}

}

* class দিয়ে ক্লাস ডিফাইন করি।
* constructor হল একটা বিশেষ মেথড, যা নতুন অবজেক্ট তৈরি হলে স্বয়ংক্রিয়ভাবে কল হয়।
* মেথডগুলো ফাংশনের মতোই কাজ করে, কিন্তু function লিখতে হয় না।

**৩. কিভাবে Class থেকে অবজেক্ট তৈরি করবেন?**

javascript

CopyEdit

const obj = new ClassName(arguments);

* new কীওয়ার্ড ব্যবহার করে আমরা ক্লাস থেকে নতুন অবজেক্ট তৈরি করি।
* constructor মেথড নিজে থেকেই রান হয়।

**৪. সহজ একটি উদাহরণ**

javascript

CopyEdit

class Person {

constructor(name, age) {

this.name = name; // name প্রোপার্টি সেট করলাম

this.age = age; // age প্রোপার্টি সেট করলাম

}

greet() {

console.log(`আমার নাম ${this.name} এবং আমি ${this.age} বছর বয়সী।`);

}

}

const p1 = new Person("রাফিয়া", ২৫);

p1.greet(); // আউটপুট: আমার নাম রাফিয়া এবং আমি ২৫ বছর বয়সী।

* this দ্বারা বোঝানো হয় ওই অবজেক্ট যেটার ওপর কাজ চলছে।
* greet() হল মেথড, যা অবজেক্টের তথ্য প্রিন্ট করে।

**৫. Class এর গুরুত্বপূর্ণ অংশ**

* **constructor:** অবজেক্ট তৈরির সময় ডাটা সেট করে।
* **this:** ক্লাসের ভিতরে অবজেক্ট নিজেকে নির্দেশ করে।
* **methods:** অবজেক্টের ফাংশন।

**৬. Class Expression**

Class কে ভ্যারিয়েবলে রাখাও যায়:

javascript

CopyEdit

const Animal = class {

constructor(type) {

this.type = type;

}

sound() {

console.log(`${this.type} আওয়াজ করে।`);

}

};

const cat = new Animal("বিড়াল");

cat.sound(); // বিড়াল আওয়াজ করে।

**৭. Getter ও Setter**

* **getter**: প্রোপার্টির মান ফেরত দেয়।
* **setter**: প্রোপার্টির মান সেট করার জন্য ব্যবহৃত হয়।

javascript

CopyEdit

class Circle {

constructor(radius) {

this.radius = radius;

}

get area() {

return Math.PI \* this.radius \* this.radius;

}

set area(value) {

console.log("Area সরাসরি সেট করা যায় না।");

}

}

const c = new Circle(5);

console.log(c.area); // 78.53981633974483

c.area = 100; // আউটপুট: Area সরাসরি সেট করা যায় না।

**৮. Static Method এবং Property**

* **static** মেথড ক্লাসের সাথে যুক্ত থাকে, অবজেক্টের সাথে নয়।
* সরাসরি ক্লাস থেকে কল করতে হয়।

javascript

CopyEdit

class MathUtil {

static multiply(a, b) {

return a \* b;

}

}

console.log(MathUtil.multiply(3, 4)); // 12

**৯. Inheritance (ওয়ারিস)**

একটা ক্লাস অন্য ক্লাস থেকে প্রোপার্টি এবং মেথড পেতে পারে extends ব্যবহার করে।

javascript

CopyEdit

class Animal {

constructor(name) {

this.name = name;

}

speak() {

console.log(`${this.name} শব্দ করে।`);

}

}

class Dog extends Animal {

constructor(name, breed) {

super(name); // প্যারেন্ট ক্লাসের constructor কল করলাম

this.breed = breed;

}

speak() {

console.log(`${this.name} ভোকে।`);

}

}

const dog1 = new Dog("ববি", "ল্যাব্রাডর");

dog1.speak(); // ববি ভোকে।

**১০. Important Notes (গুরুত্বপূর্ণ তথ্য)**

* Class হলো Prototype ভিত্তিক inheritance এর সহজ সিনট্যাক্স।
* ক্লাসের মেথডগুলো অবজেক্টে নতুন করে তৈরি হয় না, স্মৃতি বাঁচে।
* new ছাড়া Class কল করলে Error দেয়।
* একক ওয়ারিস (single inheritance) সমর্থিত, multiple inheritance নেই।
* নতুন ভার্সনে Private field (যেমন #ssn) আসে।

**১১. Private Fields (নতুন ফিচার)**

# দিয়ে প্রাইভেট প্রোপার্টি তৈরি করা যায়, যেগুলো ক্লাসের বাইরে দেখা বা ব্যবহার করা যায় না।

javascript

CopyEdit

class Student {

#rollNumber;

constructor(name, rollNumber) {

this.name = name;

this.#rollNumber = rollNumber;

}

getRollNumber() {

return this.#rollNumber;

}

}

const s = new Student("রাফিয়া", ১০১);

console.log(s.name); // রাফিয়া

// console.log(s.#rollNumber); // Error: private property

console.log(s.getRollNumber()); // ১০১

**১২. সারসংক্ষেপ টেবিল**

| **ফিচার** | **বর্ণনা** | **উদাহরণ** |
| --- | --- | --- |
| Class ঘোষনা | class ClassName {} | class Person {} |
| Constructor | অবজেক্ট ইনিশিয়ালাইজেশনের জন্য | constructor(name) {} |
| মেথড | ক্লাসের ফাংশন | greet() {} |
| Static মেথড | ক্লাসে সরাসরি কল করা হয় | static multiply(a,b) {} |
| Inheritance | অন্য ক্লাস থেকে ওয়ারিস | class Dog extends Animal {} |
| super() কল | প্যারেন্ট constructor কল | super(name) |
| Private ফিল্ড | প্রাইভেট প্রোপার্টি (#) | #rollNumber; |
| Getter / Setter | প্রোপার্টির getter/setter | get area() {}, set area() {} |

**১৩. পূর্ণাঙ্গ উদাহরণ**

javascript

CopyEdit

class Employee {

#salary; // প্রাইভেট

constructor(name, position, salary) {

this.name = name;

this.position = position;

this.#salary = salary;

}

getSalary() {

return this.#salary;

}

setSalary(amount) {

if (amount > 0) {

this.#salary = amount;

} else {

console.log("সঠিক স্যালারি দিন");

}

}

work() {

console.log(`${this.name} ${this.position} হিসাবে কাজ করছে।`);

}

static company() {

return "Tech Corp Ltd.";

}

}

const emp = new Employee("রাফিয়া", "ডেভেলপার", 50000);

emp.work(); // রাফিয়া ডেভেলপার হিসাবে কাজ করছে।

console.log(emp.getSalary()); // 50000

emp.setSalary(60000);

console.log(emp.getSalary()); // 60000

console.log(Employee.company()); // Tech Corp Ltd.

।

**🌀 JavaScript Loop (লুপ) – সম্পূর্ণ বাংলা নোট**

**🔶 ১. Loop কী?**

**Loop** হল এমন একটি প্রোগ্রামিং কৌশল, যার মাধ্যমে এক কাজ বারবার করা যায় যতক্ষণ না কোনো নির্দিষ্ট শর্ত পূরণ হয়।

📌 সহজ ভাষায় — একটি কোড ব্লক **একাধিক বার চালাতে** চাইলে লুপ ব্যবহার করা হয়।

**🔷 ২. লুপের ধরন (Types of Loop)**

JavaScript-এ প্রধানত ৫ ধরনের লুপ আছে:

| **লুপের নাম** | **কখন ব্যবহার হয়** |
| --- | --- |
| for loop | নির্দিষ্ট সংখ্যক বার চালাতে |
| while loop | শর্ত সত্য থাকলে চালাতে |
| do...while loop | কমপক্ষে একবার চালাতে হয় |
| for...in loop | অবজেক্টের property গুলো নিয়ে কাজ করতে |
| for...of loop | iterable (array/string) নিয়ে কাজ করতে |

**🔹 ৩. for loop**

**🔸 Syntax:**

javascript

CopyEdit

for (initialization; condition; increment/decrement) {

// কোড ব্লক

}

**🔸 উদাহরণ:**

javascript

CopyEdit

for (let i = 1; i <= 5; i++) {

console.log("সংখ্যা:", i);

}

🔹 আউটপুট:

makefile

CopyEdit

সংখ্যা: 1

সংখ্যা: 2

সংখ্যা: 3

সংখ্যা: 4

সংখ্যা: 5

**🔹 ৪. while loop**

**🔸 Syntax:**

javascript

CopyEdit

while (condition) {

// কোড ব্লক

}

**🔸 উদাহরণ:**

javascript

CopyEdit

let i = 1;

while (i <= 5) {

console.log("While Loop:", i);

i++;

}

🔹 আউটপুট:

vbnet

CopyEdit

While Loop: 1

While Loop: 2

While Loop: 3

While Loop: 4

While Loop: 5

**🔹 ৫. do...while loop**

➡️ এই লুপে কোড **কমপক্ষে একবার চালায়**, তারপর শর্ত চেক করে।

**🔸 Syntax:**

javascript

CopyEdit

do {

// কোড ব্লক

} while (condition);

**🔸 উদাহরণ:**

javascript

CopyEdit

let i = 1;

do {

console.log("Do While Loop:", i);

i++;

} while (i <= 5);

🔹 আউটপুট:

vbnet

CopyEdit

Do While Loop: 1

Do While Loop: 2

Do While Loop: 3

Do While Loop: 4

Do While Loop: 5

**🔹 ৬. for...in loop**

➡️ এই লুপ **object** এর property গুলোর উপর চালানো হয়।

**🔸 উদাহরণ:**

javascript

CopyEdit

const student = { name: "Rafia", age: 22, subject: "JavaScript" };

for (let key in student) {

console.log(key + ":", student[key]);

}

🔹 আউটপুট:

makefile

CopyEdit

name: Rafia

age: 22

subject: JavaScript

**🔹 ৭. for...of loop**

➡️ এই লুপ array, string, set, map ইত্যাদি **iterable** এর উপর চালানো হয়।

**🔸 উদাহরণ:**

javascript

CopyEdit

const fruits = ["Apple", "Banana", "Mango"];

for (let fruit of fruits) {

console.log("ফল:", fruit);

}

🔹 আউটপুট:

makefile

CopyEdit

ফল: Apple

ফল: Banana

ফল: Mango

**🔸 ৮. লুপে break এবং continue**

**🔹 break: লুপ থামিয়ে দেয়।**

**🔹 continue: লুপের ওই ধাপ skip করে, পরের ধাপে যায়।**

javascript

CopyEdit

for (let i = 1; i <= 5; i++) {

if (i === 3) {

continue; // 3 skip করবে

}

console.log(i);

}

🔹 আউটপুট:

CopyEdit

1

2

4

5

javascript

CopyEdit

for (let i = 1; i <= 5; i++) {

if (i === 3) {

break; // এখানেই লুপ থেমে যাবে

}

console.log(i);

}

🔹 আউটপুট:

CopyEdit

1

2

**✅ সারাংশ (Summary)**

| **লুপ** | **কখন ব্যবহার করবেন** | **শর্ত** |
| --- | --- | --- |
| for | নির্দিষ্ট সংখ্যক বার | আগে চেক |
| while | শর্ত সত্য থাকলে | আগে চেক |
| do...while | একবার অন্তত চালাতে চাইলে | পরে চেক |
| for...in | object এর property ঘোরাতে | key |
| for...of | array/string এর উপর | value |

**👩‍🏫 JavaScript Subclass / Inheritance (ওয়ারিশ/উত্তরাধিকার) – বাংলা নোট**

**🔷 ১. Subclass মানে কী?**

JavaScript-এ যখন একটা **class অন্য একটা class থেকে প্রোপার্টি (property) এবং মেথড (method)** গ্রহণ করে, তখন তাকে বলে **subclass** বা **child class**। যেটা থেকে গ্রহণ করছে, সেটাকে বলে **superclass** বা **parent class**।

📌 সহজ কথায় — subclass মানে হলো **একটা নতুন class তৈরি করা**, যেটা আগের class-এর সব কিছু **উত্তরাধিকারসূত্রে পায়**।

**🔷 ২. কীভাবে subclass তৈরি করা হয়?**

Subclass তৈরি করতে extends এবং super() ব্যবহার করতে হয়।

**✅ Syntax:**

javascript

CopyEdit

class ParentClass {

constructor() {

// কোড

}

parentMethod() {

// কোড

}

}

class ChildClass extends ParentClass {

constructor() {

super(); // parent constructor কল করে

// অতিরিক্ত কোড

}

childMethod() {

// কোড

}

}

**🧠 Keyword বোঝো:**

| **কীওয়ার্ড** | **কাজ** |
| --- | --- |
| extends | এক class কে অন্য class থেকে উত্তরাধিকার দেয় |
| super() | parent class-এর constructor/method কল করতে ব্যবহার হয় |
| this | বর্তমান অবজেক্টকে বোঝায় |

**🔷 ৩. সহজ উদাহরণ:**

javascript

CopyEdit

// Parent class

class Animal {

constructor(name) {

this.name = name;

}

speak() {

console.log(`${this.name} আওয়াজ করছে`);

}

}

// Child class (subclass)

class Dog extends Animal {

constructor(name, breed) {

super(name); // Animal এর constructor কল

this.breed = breed;

}

bark() {

console.log(`${this.name} ঘেউ ঘেউ করছে (${this.breed})`);

}

}

const myDog = new Dog("ব্রুনি", "ল্যাব্রাডর");

myDog.speak(); // ব্রুনি আওয়াজ করছে

myDog.bark(); // ব্রুনি ঘেউ ঘেউ করছে (ল্যাব্রাডর)

**🔷 ৪. subclass এর সুবিধা**

✅ Reusability (বারবার ব্যবহারযোগ্য)  
✅ কোড ছোট হয়  
✅ Parent class-এর সব ফিচার subclass পায়  
✅ নতুন ফিচার subclass-এ আলাদাভাবে যোগ করা যায়

**🔷 ৫. Method Override (প্যারেন্টের method পরিবর্তন করা)**

Subclass চাইলে parent class-এর method **override** করতে পারে।

javascript

CopyEdit

class Animal {

speak() {

console.log("কোনো প্রাণী আওয়াজ করছে");

}

}

class Cat extends Animal {

speak() {

console.log("বিড়াল মিউ মিউ করছে");

}

}

const cat = new Cat();

cat.speak(); // বিড়াল মিউ মিউ করছে

➡️ এখানে speak() প্যারেন্ট ক্লাসেও ছিল, কিন্তু Cat ক্লাস নিজস্বভাবে এটাকে override করেছে।

**🔷 ৬. Static মেথড Inheritance**

javascript

CopyEdit

class Parent {

static sayHi() {

console.log("Hi from Parent");

}

}

class Child extends Parent {}

Child.sayHi(); // Hi from Parent

➡️ Static method ও subclass ইনহেরিট করে নিতে পারে।

**🔷 ৭. instanceof দিয়ে চেক করা**

javascript

CopyEdit

console.log(myDog instanceof Dog); // true

console.log(myDog instanceof Animal); // true

**✅ সংক্ষিপ্ত টেবিল**

| **উপাদান** | **কাজ** |
| --- | --- |
| extends | Subclass তৈরি করে |
| super() | Parent class-এর constructor/method কল করে |
| this | নিজের অবজেক্ট বোঝায় |
| Method override | প্যারেন্ট মেথড পরিবর্তন করা যায় |
| Static method | class level method, subclass পায় |

**🔷 ৮. আরেকটি পূর্ণাঙ্গ উদাহরণ**

javascript

CopyEdit

class Vehicle {

constructor(name) {

this.name = name;

}

start() {

console.log(`${this.name} চালু হয়েছে`);

}

}

class Car extends Vehicle {

constructor(name, brand) {

super(name);

this.brand = brand;

}

info() {

console.log(`গাড়ির নাম: ${this.name}, ব্র্যান্ড: ${this.brand}`);

}

}

const myCar = new Car("Civic", "Honda");

myCar.start(); // Civic চালু হয়েছে

myCar.info(); // গাড়ির নাম: Civic, ব্র্যান্ড: Honda

**✨ উপসংহার:**

🔹 Subclass হল একটি class যা অন্য class থেকে গুণাবলি গ্রহণ করে।  
🔹 এতে কোডের পুনরাবৃত্তি কম হয় এবং সহজে manage করা যায়।  
🔹 extends দিয়ে subclass তৈরি করা হয়, আর super() দিয়ে parent এর constructor কল করা হয়।

**🔷 ১. Constructor কী?**

🔹 constructor হলো একটি **বিশেষ মেথড**, যা **class থেকে নতুন অবজেক্ট তৈরি হলে স্বয়ংক্রিয়ভাবে (Automatically) কল হয়**।

📌 এটি মূলত object-এর **initial value সেট করার কাজ করে**, যেমন: name, age, type ইত্যাদি।

**✅ Syntax:**

javascript

CopyEdit

class ClassName {

constructor(parameters) {

// প্রোপার্টি initialize করা হয়

this.property = value;

}

}

**🔷 ২. একটি সহজ উদাহরণ:**

javascript

CopyEdit

class Person {

constructor(name, age) {

this.name = name;

this.age = age;

}

introduce() {

console.log(`আমার নাম ${this.name} এবং আমার বয়স ${this.age} বছর।`);

}

}

const p1 = new Person("রাফিয়া", ২২);

p1.introduce(); // আমার নাম রাফিয়া এবং আমার বয়স ২২ বছর।

**🔎 ব্যাখ্যা:**

* constructor(name, age) হল সেই ফাংশন যেটা object তৈরি সময় কল হয়।
* this.name এবং this.age মান গুলো object-এর প্রপার্টি হিসেবে সেট হয়।

**🔷 ৩. constructor এর বৈশিষ্ট্য**

| **বৈশিষ্ট্য** | **ব্যাখ্যা** |
| --- | --- |
| শুধুমাত্র একটি constructor থাকে | এক ক্লাসে একটাই constructor থাকতে পারে |
| object তৈরি হলে নিজে নিজে কল হয় | new ClassName() দিলে constructor নিজে চালু হয় |
| this keyword ব্যবহার করে প্রোপার্টি সেট করা হয় | this মানে হলো বর্তমান অবজেক্ট |

**🔷 ৪. Default constructor (যদি না দেওয়া হয়)**

যদি তুমি constructor না লেখো, JavaScript **অটোমেটিক** একটা ফাঁকা constructor তৈরি করে নেয়।

javascript

CopyEdit

class Student {}

const s = new Student(); // কোনো error হবে না

**🔷 ৫. Constructor সহ Inheritance (প্যারেন্ট-চাইল্ড ক্লাসে)**

Subclass-এ constructor ব্যবহার করলে super() দিয়ে প্যারেন্ট constructor কল করতেই হবে।

javascript

CopyEdit

class Animal {

constructor(name) {

this.name = name;

}

}

class Dog extends Animal {

constructor(name, breed) {

super(name); // প্যারেন্ট constructor কল করা হল

this.breed = breed;

}

info() {

console.log(`${this.name} একটি ${this.breed} জাতের কুকুর।`);

}

}

const d = new Dog("ব্রুনি", "ল্যাব্রাডর");

d.info(); // ব্রুনি একটি ল্যাব্রাডর জাতের কুকুর।

**🔷 ৬. Multiple Parameters (একাধিক ইনপুট নেওয়া)**

javascript

CopyEdit

class Product {

constructor(name, price, quantity) {

this.name = name;

this.price = price;

this.quantity = quantity;

}

totalPrice() {

return this.price \* this.quantity;

}

}

const p = new Product("Laptop", 50000, 2);

console.log(p.totalPrice()); // 100000

**🔷 ৭. Arrow Function ব্যবহার constructor-এ**

Constructor এর ভিতরে তুমি arrow function ব্যবহার করে মেথড বানাতে পারো:

javascript

CopyEdit

class Counter {

constructor() {

this.count = 0;

this.increment = () => {

this.count++;

};

}

}

const c = new Counter();

c.increment();

c.increment();

console.log(c.count); // 2

**✅ উপসংহার:**

| **বিষয়** | **তথ্য** |
| --- | --- |
| constructor কি | class-এর special method |
| কখন চালু হয় | যখন নতুন object তৈরি হয় |
| কি করে | প্রোপার্টি initialize করে |
| subclass এ super() দরকার? | হ্যাঁ, constructor থাকলে অবশ্যই দরকার |
| কয়টা constructor থাকা যায় | একটাই |

**📘 সারাংশ:**

javascript

CopyEdit

class Example {

constructor(x, y) {

this.x = x;

this.y = y;

}

show() {

console.log(`x = ${this.x}, y = ${this.y}`);

}

}

const e = new Example(5, 10);

e.show(); // x = 5, y = 10

**🔷 ১. Getter & Setter কী?**

🔹 getter এবং setter হল **special method** যা object বা class-এর প্রপার্টির মান **পড়া (get)** এবং **সেট (set)** করার জন্য ব্যবহার করা হয়।

✅ এগুলো দেখতে ফাংশনের মতো হলেও, ব্যবহার হয় **প্রপার্টির মতো**  
✅ ব্যবহারকারী বুঝতেই পারে না, সে ফাংশন কল করছে না – মনে হয় প্রপার্টির সাথে কাজ করছে।

**🔶 ২. Getter কী?**

🔸 **get keyword** দিয়ে বানানো হয়।  
🔸 কোনো প্রপার্টির **মান রিটার্ন** করতে ব্যবহৃত হয়।

javascript

CopyEdit

class Student {

constructor(name, marks) {

this.name = name;

this.marks = marks;

}

get result() {

return this.marks >= 33 ? "পাস" : "ফেল";

}

}

const s1 = new Student("রাফিয়া", 45);

console.log(s1.result); // আউটপুট: পাস

এখানে result() এর মত কল করা লাগছে না। শুধু s1.result লিখলেই কাজ হচ্ছে।

**🔶 ৩. Setter কী?**

🔸 **set keyword** দিয়ে বানানো হয়।  
🔸 কোনো প্রপার্টির **মান আপডেট বা কাস্টমভাবে সেট** করতে ব্যবহৃত হয়।

javascript

CopyEdit

class Product {

constructor(name, price) {

this.name = name;

this.\_price = price;

}

get price() {

return this.\_price + " টাকা";

}

set price(value) {

if (value < 0) {

console.log("দয়া করে সঠিক দাম দিন!");

} else {

this.\_price = value;

}

}

}

const p = new Product("Mobile", 15000);

console.log(p.price); // 15000 টাকা

p.price = -500; // দয়া করে সঠিক দাম দিন!

p.price = 18000;

console.log(p.price); // 18000 টাকা

এখানে \_price নাম দিয়ে মূল ভ্যালুটা রাখা হয়েছে, কারণ price নামটাই getter/setter হিসেবে ব্যবহার হচ্ছে।

**🔸 কেন \_price লেখা হলো?**

➡️ এটা হলো **naming convention**।  
➡️ আমরা যদি get price() এবং set price() ব্যবহার করি, তখন price নামের একটা আলাদা ভ্যারিয়েবল রাখা ঠিক হবে না। তাই \_price নামে ভ্যারিয়েবল রাখা হয়, যাতে conflict না হয়।

**🔷 ৪. Getter & Setter object literal এ**

javascript

CopyEdit

const person = {

firstName: "Rafia",

lastName: "Akter",

get fullName() {

return this.firstName + " " + this.lastName;

},

set fullName(name) {

const parts = name.split(" ");

this.firstName = parts[0];

this.lastName = parts[1];

}

};

console.log(person.fullName); // Rafia Akter

person.fullName = "Nusrat Jahan";

console.log(person.firstName); // Nusrat

console.log(person.lastName); // Jahan

**🔷 ৫. উপকারিতা**

| **উপকারিতা** | **ব্যাখ্যা** |
| --- | --- |
| কোডকে সহজভাবে পড়া যায় | getter/setter দেখে প্রপার্টির মতো লাগে |
| ডেটা validate করা যায় | setter-এ শর্ত বসানো যায় |
| encapsulation সহজ হয় | অভ্যন্তরীণ ডেটা নিয়ন্ত্রণ সহজ হয় |

**🧠 সংক্ষেপে মনে রাখার নিয়ম:**

| **Keyword** | **কাজ** | **কিভাবে ব্যবহার হয়** |
| --- | --- | --- |
| get | প্রপার্টি রিড করা | obj.prop |
| set | প্রপার্টি সেট করা | obj.prop = value |

**🔚 উপসংহার:**

* Getter দিয়ে প্রপার্টির ভ্যালু "রিডেবল" করা যায়।
* Setter দিয়ে ভ্যালু কন্ট্রোল/validate করে সেট করা যায়।
* নামের আগে \_ দিলে বোঝা যায় এটা private/internal variable।
* Getter/Setter প্রপার্টির মতোই ব্যবহার হয়, ফাংশনের মতো নয়।

**📌 Extra Tip:**

👉 Getter/Setter দিয়ে **read-only প্রপার্টিও** বানানো যায়।  
যদি শুধু get থাকে, তাহলে সেটার মান পরিবর্তন করা যাবে না।

**🔷 ১. JavaScript কীভাবে কোড চালায়?**

JavaScript একধরনের **single-threaded** এবং **non-blocking** ভাষা।

➡️ অর্থাৎ, এটি একসাথে একটি কাজই করে, কিন্তু সেই কাজ **পিছনে রেখে পরবর্তী কাজ** চালিয়ে যেতে পারে – যেটা **Asynchronous** পদ্ধতির কারণে সম্ভব।

**🔶 ২. Synchronous (সিঙ্ক্রোনাস) কী?**

📌 Synchronous মানে:  
**একটা কাজ শেষ না হওয়া পর্যন্ত পরবর্তী কাজ শুরু হবে না।**

➡️ এটি লাইন বাই লাইন চলে (top to bottom)।  
➡️ যদি কোনো লাইনে দেরি হয়, তাহলে পরবর্তী লাইন অপেক্ষা করে।

**🔸 উদাহরণ:**

javascript

CopyEdit

console.log("১. শুরু");

console.log("২. মাঝখান");

console.log("৩. শেষ");

🔹 আউটপুট:

CopyEdit

১. শুরু

২. মাঝখান

৩. শেষ

➡️ সব কাজ **ধাপে ধাপে, লাইন বাই লাইন** সম্পন্ন হয়েছে।

**🔶 ৩. Asynchronous (এসিঙ্ক্রোনাস) কী?**

📌 Asynchronous মানে:  
**যে কাজগুলো সময়সাপেক্ষ, সেগুলোকে পেছনে পাঠিয়ে অন্য কাজগুলো আগেই শেষ করে ফেলা।**

➡️ সময় লাগবে এমন কাজ (যেমন API call, File Load, Timer) জাভাস্ক্রিপ্ট আগে ট্রিগার করে, কিন্তু **ব্লক না করে** পরবর্তী লাইনে চলে যায়।

**🔸 উদাহরণ (setTimeout ব্যবহার করে):**

javascript

CopyEdit

console.log("১. শুরু");

setTimeout(() => {

console.log("২. অপেক্ষার পর (৩ সেকেন্ড)");

}, 3000);

console.log("৩. শেষ");

🔹 আউটপুট:

scss

CopyEdit

১. শুরু

৩. শেষ

২. অপেক্ষার পর (৩ সেকেন্ড)

➡️ এখানে setTimeout() পেছনে চলে গেছে এবং **JavaScript পরবর্তী কাজগুলো আগেই করে ফেলেছে।**

**🔷 ৪. তুলনামূলক পার্থক্য (Synchronous vs Asynchronous)**

| **বৈশিষ্ট্য** | **Synchronous** | **Asynchronous** |
| --- | --- | --- |
| কাজের ধরণ | একটার পর একটা | পেছনের কাজ চলতে চলতে সামনের কাজ |
| Line execution | লাইন বাই লাইন | ডিলে কাজ পেছনে পাঠায় |
| Performance impact | ব্লক করতে পারে | ব্লক না করে smooth performance |
| উদাহরণ | console.log() | setTimeout(), fetch(), API call |

**🔶 ৫. কখন Synchronous, কখন Asynchronous?**

| **Task** | **ধরণ** |
| --- | --- |
| Variable declaration | Synchronous |
| Simple function call | Synchronous |
| setTimeout / setInterval | Asynchronous |
| API fetch (network request) | Asynchronous |
| File read/write | Asynchronous |
| Promise / async-await | Asynchronous |

**🔷 ৬. বাস্তব উদাহরণ:**

**🔸 বাস্তব জীবন: Synchronous**

1. দোকানে গিয়ে একটার পর একটা জিনিস কিনছো, পরেরটা শুরু করছো আগেরটা শেষ হলে।

**🔸 বাস্তব জীবন: Asynchronous**

1. খাবার অর্ডার দিয়ে পাশে দাঁড়িয়ে আরেকটা কাজ করছো, খাবার তৈরি হলে কল দিয়ে ডাকছে।

**🔷 ৭. Asynchronous কোড লেখার উপায়**

JavaScript-এ আমরা Asynchronous কোড লিখতে পারি:

1. ✅ setTimeout, setInterval
2. ✅ Promises
3. ✅ async / await

**🔸 Promises Example:**

javascript

CopyEdit

const promise = new Promise((resolve, reject) => {

setTimeout(() => {

resolve("ডেটা চলে এসেছে!");

}, 2000);

});

promise.then((data) => {

console.log(data); // ডেটা চলে এসেছে!

});

**🔸 async / await Example:**

javascript

CopyEdit

function waitData() {

return new Promise(resolve => {

setTimeout(() => resolve("ডেটা প্রস্তুত"), 2000);

});

}

async function showData() {

console.log("প্রসেস শুরু");

const result = await waitData();

console.log(result); // ডেটা প্রস্তুত

console.log("প্রসেস শেষ");

}

showData();

**✅ উপসংহার:**

| **বিষয়** | **Synchronous** | **Asynchronous** |
| --- | --- | --- |
| কাজ করে | একটার পর একটা | পিছনের কাজ রেখে সামনে এগোয় |
| ব্লক করে? | হ্যাঁ | না |
| ব্যবহার হয় | সাধারণ কোড | API, File, Timer ইত্যাদিতে |
| দ্রুত? | ধীরে ধীরে | অনেক সময় দ্রুত ফল দেয় |

📘 তোমার জন্য সংক্ষিপ্তভাবে মনে রাখার টিপ:

🔹 Synchronous = ধাপে ধাপে (line by line)  
🔹 Asynchronous = একসাথে অনেক কিছু, কিন্তু স্মার্টভাবে

**🧾 JavaScript Array Methods – বাংলায় পূর্ণাঙ্গ নোট**

**✅ ১. push() → শেষে নতুন উপাদান যোগ করে**

javascript

CopyEdit

const fruits = ["Apple", "Banana"];

fruits.push("Mango");

console.log(fruits); // ["Apple", "Banana", "Mango"]

**✅ ২. pop() → শেষ উপাদান মুছে ফেলে**

javascript

CopyEdit

const fruits = ["Apple", "Banana", "Mango"];

fruits.pop();

console.log(fruits); // ["Apple", "Banana"]

**✅ ৩. concat() → একাধিক array জোড়া দেয়**

javascript

CopyEdit

const a = [1, 2];

const b = [3, 4];

const result = a.concat(b);

console.log(result); // [1, 2, 3, 4]

**✅ ৪. fill() → পুরো array-তে একই মান বসিয়ে দেয়**

javascript

CopyEdit

const arr = [1, 2, 3, 4];

arr.fill(0);

console.log(arr); // [0, 0, 0, 0]

🔹 নির্দিষ্ট অংশে fill করতে পারো:

javascript

CopyEdit

arr.fill(5, 1, 3); // index 1 থেকে 3-র আগ পর্যন্ত

**✅ ৫. filter() → শর্ত অনুযায়ী উপাদান বেছে নেয়**

javascript

CopyEdit

const nums = [1, 2, 3, 4, 5];

const even = nums.filter(n => n % 2 === 0);

console.log(even); // [2, 4]

**✅ ৬. indexOf() → উপাদান কোন index-এ আছে**

javascript

CopyEdit

const items = ["Pen", "Book", "Pencil"];

console.log(items.indexOf("Book")); // 1

console.log(items.indexOf("Bag")); // -1 (না থাকলে)

**✅ ৭. join() → array কে string বানায়**

javascript

CopyEdit

const names = ["Rafi", "Sami", "Tina"];

console.log(names.join(", ")); // "Rafi, Sami, Tina"

**✅ ৮. reverse() → উল্টো করে**

javascript

CopyEdit

const nums = [1, 2, 3];

nums.reverse();

console.log(nums); // [3, 2, 1]

**✅ ৯. slice(start, end) → অংশ কেটে নেয় (মূল array বদলায় না)**

javascript

CopyEdit

const arr = [10, 20, 30, 40, 50];

const sliced = arr.slice(1, 4);

console.log(sliced); // [20, 30, 40]

**✅ ১০. splice(start, deleteCount, item...)**

🔸 কিছু মুছে ফেলে এবং চাইলে কিছু যোগও করে (মূল array বদলে যায়)

javascript

CopyEdit

const arr = [1, 2, 3, 4, 5];

arr.splice(2, 1); // index 2 থেকে 1 উপাদান মুছে ফেলে

console.log(arr); // [1, 2, 4, 5]

🔸 যোগ করতেও পারে:

javascript

CopyEdit

arr.splice(2, 0, 99); // index 2 তে 99 যোগ করো, কিছু মুছো না

console.log(arr); // [1, 2, 99, 4, 5]

**✅ ১১. sort() → উপাদানকে সাজায় (string হিসেবে)**

javascript

CopyEdit

const items = [4, 10, 2, 1];

items.sort();

console.log(items); // [1, 10, 2, 4] → ভুল কারণ string sort

🔹 ঠিকভাবে সংখ্যা sort করতে:

javascript

CopyEdit

items.sort((a, b) => a - b); // Ascending

items.sort((a, b) => b - a); // Descending

**✅ ১২. Array-তে নতুন উপাদান যোগ (Add)**

🔸 প্রথমে যোগ করতে → unshift()  
🔸 শেষে যোগ করতে → push()

javascript

CopyEdit

const arr = [2, 3];

arr.unshift(1); // শুরুতে যোগ

arr.push(4); // শেষে যোগ

console.log(arr); // [1, 2, 3, 4]

**🧠 সংক্ষিপ্ত টেবিল:**

| **Method** | **কাজ** |
| --- | --- |
| push() | শেষে যোগ করে |
| pop() | শেষে থেকে সরায় |
| unshift() | শুরুতে যোগ করে |
| shift() | শুরু থেকে সরায় |
| concat() | একাধিক array যোগ করে |
| fill() | পুরো array-তে একই মান বসায় |
| filter() | শর্ত অনুযায়ী বাছাই করে |
| indexOf() | উপাদান খুঁজে index দেয় |
| join() | array কে string বানায় |
| reverse() | উল্টো করে |
| slice() | নির্দিষ্ট অংশ কেটে নেয় |
| splice() | কেটে, মুছে বা যোগ করে |
| sort() | সাজায় (সংখ্যার জন্য কম্পেয়ার ফাংশন দরকার) |

**✅ ১. Function কী?**

🔹 **Function** হলো এক ধরনের কোড ব্লক, যেটা কোনো নির্দিষ্ট কাজ করার জন্য তৈরি করা হয়।  
🔹 একবার তৈরি করলে সেটা বারবার ব্যবহার করা যায়।

📌 সহজ ভাষায় – function হচ্ছে **reusable code block**।

**🔶 ২. Function কিভাবে তৈরি করা হয়?**

**🔸 Syntax (Function Declaration):**

javascript

CopyEdit

function functionName(parameters) {

// কাজ

}

**🔸 Example:**

javascript

CopyEdit

function sayHello() {

console.log("হ্যালো দুনিয়া!");

}

📌 এখানে sayHello() নামের একটি ফাংশন তৈরি হয়েছে।

**✅ ৩. Function Call করা (Function Add)**

👉 একবার ফাংশন তৈরি করে নিলে, নিচের মতো করে কল করা হয়:

javascript

CopyEdit

sayHello(); // হ্যালো দুনিয়া!

**✅ ৪. Function with Parameters (মান পাঠানো)**

javascript

CopyEdit

function greet(name) {

console.log("হ্যালো, " + name);

}

greet("রাফিয়া"); // হ্যালো, রাফিয়া

greet("রহিম"); // হ্যালো, রহিম

➡️ এখানে name হলো parameter, আর "রাফিয়া" বা "রহিম" হলো argument।

**✅ ৫. Function Return – মান ফেরত পাঠানো**

🔸 কোনো ফাংশনের মাধ্যমে আমরা একটা মান **return** করতে পারি।

javascript

CopyEdit

function add(a, b) {

return a + b;

}

const result = add(5, 3);

console.log("ফল:", result); // ফল: 8

➡️ এখানে add() ফাংশন দুটি সংখ্যার যোগফল **return** করছে।

**✅ ৬. Return মান ব্যাখ্যা**

| **বিষয়** | **ব্যাখ্যা** |
| --- | --- |
| return কি | ফাংশন থেকে কোনো মান ফিরিয়ে দেয় |
| return দিলে কী হয়? | ফাংশন থেমে যায় এবং মান ফিরে আসে |
| return না দিলে কী হয়? | ফাংশন undefined রিটার্ন করে |

**✅ ৭. Return ছাড়া ফাংশন**

javascript

CopyEdit

function multiply(x, y) {

console.log(x \* y);

}

multiply(2, 5); // 10

➡️ এখানে console.log() করেছে, কিন্তু কোনো মান return করেনি।

**✅ ৮. Function Expression (ভেরিয়েবল দিয়ে ফাংশন)**

javascript

CopyEdit

const sayBye = function() {

console.log("বিদায়!");

};

sayBye(); // বিদায়!

**✅ ৯. Arrow Function (ছোট করে লেখার উপায়)**

javascript

CopyEdit

const square = (n) => {

return n \* n;

};

console.log(square(4)); // 16

📌 যদি return ছাড়া ছোট ফাংশন হয়:

javascript

CopyEdit

const double = n => n \* 2;

console.log(double(5)); // 10

**🧠 সংক্ষিপ্ত টেবিল:**

| **বিষয়** | **উদাহরণ** |
| --- | --- |
| Function তৈরি | function add(a, b) {} |
| Call করা | add(5, 6); |
| Parameter পাঠানো | greet("Rafia"); |
| Return করা | return a + b; |
| Arrow Function | const square = x => x \* x; |

**🧪 চর্চার জন্য কয়েকটি ছোট ফাংশন:**

javascript

CopyEdit

// ১. দুই নাম্বারের গুণফল

function multiply(a, b) {

return a \* b;

}

// ২. সংখ্যা বিজোড় নাকি জোড়?

function isEven(n) {

return n % 2 === 0;

}

// ৩. বর্গফল

const square = n => n \* n;

**✅ উপসংহার:**

* Function হলো নির্দিষ্ট কাজের জন্য কোড ব্লক
* Function Call দিলে কাজ শুরু হয়
* Return দিলে মান ফিরে আসে, নাহলে undefined
* Arrow function ছোট করে লেখার পদ্ধতি

**👨‍💻 JavaScript OOP (Object-Oriented Programming) – সম্পূর্ণ বাংলা নোট**

**🧠 OOP মানে কী?**

📌 **OOP (অবজেক্ট ভিত্তিক প্রোগ্রামিং)** হলো এমন একটি প্রোগ্রামিং ধারণা যেখানে আমরা কোডকে অবজেক্ট আকারে সাজাই, যেনো রিয়েল লাইফ জিনিসের মতো ব্যবহার করা যায়।

JavaScript একটি **object-based language** — তাই OOP ব্যবহার করেই আমরা বড় বড় প্রজেক্ট সহজে ও গুছিয়ে করতে পারি।

**🎯 OOP-এর ৪টি মূল বৈশিষ্ট্য (Pillars of OOP)**

| **বৈশিষ্ট্য** | **ব্যাখ্যা** |
| --- | --- |
| 1️⃣ **Encapsulation** | ডেটা ও মেথডকে একসাথে লুকানো |
| 2️⃣ **Abstraction** | অপ্রয়োজনীয় জিনিস লুকানো |
| 3️⃣ **Inheritance** | এক ক্লাস থেকে আরেক ক্লাসে বৈশিষ্ট্য নেওয়া |
| 4️⃣ **Polymorphism** | এক কাজকে ভিন্নভাবে করা |

**🔸 ১. Object ও Class কী?**

**✅ Class:**

**Class** হলো একটি ব্লুপ্রিন্ট বা ছাঁচ — যেটা দিয়ে object তৈরি হয়।

javascript

CopyEdit

class Person {

constructor(name, age) {

this.name = name;

this.age = age;

}

speak() {

console.log(`আমি ${this.name}, বয়স ${this.age} বছর`);

}

}

**✅ Object:**

Class থেকে বানানো instance-কে বলে **object**।

javascript

CopyEdit

const p1 = new Person("রাফিয়া", 22);

p1.speak(); // আমি রাফিয়া, বয়স 22 বছর

**🔸 ২. Encapsulation (ঘেরাও করে রাখা)**

➡️ Data আর তার কাজ (method) একসাথে class-এর মধ্যে রাখা হয় = encapsulation।

javascript

CopyEdit

class BankAccount {

constructor(owner, balance) {

this.owner = owner;

this.\_balance = balance; // underscore দিয়ে private বোঝানো হয়

}

getBalance() {

return this.\_balance;

}

deposit(amount) {

if (amount > 0) {

this.\_balance += amount;

}

}

}

➡️ এখানে balance সরাসরি access করা যাবে না। Getter/setter দিয়ে করতে হবে।

**🔸 ৩. Inheritance (উত্তরাধিকার)**

➡️ এক class থেকে আরেক class সব কিছু **উত্তরাধিকার সূত্রে** পায়।

javascript

CopyEdit

class Animal {

constructor(name) {

this.name = name;

}

speak() {

console.log(`${this.name} আওয়াজ করছে`);

}

}

class Dog extends Animal {

constructor(name, breed) {

super(name);

this.breed = breed;

}

bark() {

console.log(`${this.name} ঘেউ ঘেউ করে (${this.breed})`);

}

}

const dog1 = new Dog("ব্রুনি", "ল্যাব্রাডর");

dog1.speak(); // ব্রুনি আওয়াজ করছে

dog1.bark(); // ব্রুনি ঘেউ ঘেউ করে (ল্যাব্রাডর)

**🔸 ৪. Polymorphism (এক কাজের একাধিক রূপ)**

➡️ একই method বিভিন্ন class-এ ভিন্নভাবে কাজ করে।

javascript

CopyEdit

class Animal {

speak() {

console.log("প্রাণী আওয়াজ করছে");

}

}

class Cat extends Animal {

speak() {

console.log("বিড়াল মিউ মিউ করে");

}

}

const c = new Cat();

c.speak(); // বিড়াল মিউ মিউ করে

➡️ এখানে speak() method parent আর child class দুটোতেই ভিন্ন কাজ করছে।

**🔸 ৫. Abstraction (অপ্রয়োজনীয় জিনিস লুকানো)**

➡️ ইউজারকে ডেটা বা কাজের ভিতরের গঠন না জানিয়ে, শুধু কাজের ফলাফল দেখানো।

javascript

CopyEdit

class Car {

startEngine() {

this.#privateStart(); // ভিতরের কাজ দেখা যাচ্ছে না

console.log("Engine চালু হলো");

}

#privateStart() {

console.log("গোপন ইঞ্জিন ফাংশন");

}

}

const myCar = new Car();

myCar.startEngine();

// আউটপুট:

// গোপন ইঞ্জিন ফাংশন

// Engine চালু হলো

➡️ # দিয়ে শুরু করলে সেটি private মেথড হয়।

**🧪 Practice OOP in JavaScript:**

javascript

CopyEdit

class Student {

constructor(name, roll) {

this.name = name;

this.roll = roll;

}

info() {

return `${this.name} এর রোল নাম্বার ${this.roll}`;

}

}

const s1 = new Student("রাফিয়া", 101);

console.log(s1.info()); // রাফিয়া এর রোল নাম্বার 101

**✅ OOP vs Procedural**

| **বিষয়** | **OOP** | **Procedural** |
| --- | --- | --- |
| Code Structure | Object-based | Function-based |
| Reusability | অনেক বেশি | তুলনামূলক কম |
| Maintainability | সহজ | কঠিন |
| উদাহরণ | class, object, method | function, if, loop |

**🔚 উপসংহার:**

* **OOP**-এ সবকিছু ঘোরে class ও object কে ঘিরে
* OOP আমাদের কোডকে করে:  
  ✅ পুনর্ব্যবহারযোগ্য  
  ✅ সুন্দর ও গুছানো  
  ✅ বড় প্রজেক্টে maintainable

**🔖 Extra: Built-in JS OOP**

JavaScript-এ সব কিছু অবজেক্ট, এমনকি:

javascript

CopyEdit

let x = new String("hello"); // object

let y = new Number(123); // object

কিন্তু সাধারণত আমরা ব্যবহার করি:

javascript

CopyEdit

let str = "hello"; // primitive

**🧱 JavaScript DOM (Document Object Model) – বাংলা নোট**

**🔷 ১. DOM কী?**

📌 DOM বা Document Object Model হলো:

**একটি ওয়েবপেজের HTML Structure-এর JavaScript Representation**, যা একটা **tree-structure** আকারে থাকে, এবং JavaScript দিয়ে এটা **পড়া, পরিবর্তন (change), যোগ (add), বা মুছে (delete)** ফেলা যায়।

**🔷 ২. DOM Tree Structure কেমন?**

HTML কোড DOM-এর মধ্যে **নোড (Node)** বা **অবজেক্টে** পরিণত হয়।

html

CopyEdit

<body>

<h1>Hello</h1>

<p>This is a paragraph.</p>

</body>

➡️ এই কোড DOM Tree-তে রূপ নেয়:

css

CopyEdit

document

└── html

└── body

├── h1

└── p

**🔷 ৩. DOM-এর মাধ্যমে কী কী করা যায়?**

✅ HTML Element Access করা  
✅ Content Change করা  
✅ CSS Style Change করা  
✅ New Element Add করা  
✅ Element Delete করা  
✅ User Input নিয়ে কাজ করা

**🔶 ৪. DOM Access করার পদ্ধতি (Selectors)**

| **Method** | **কাজ করে** |
| --- | --- |
| getElementById() | ID দিয়ে খুঁজে |
| getElementsByClassName() | Class দিয়ে খুঁজে (HTMLCollection) |
| getElementsByTagName() | Tag দিয়ে খুঁজে (HTMLCollection) |
| querySelector() | CSS Selector দিয়ে একটি element |
| querySelectorAll() | CSS Selector দিয়ে একাধিক element (NodeList) |

**✅ উদাহরণ:**

html

CopyEdit

<p id="demo">Hello</p>

javascript

CopyEdit

const p = document.getElementById("demo");

console.log(p.innerText); // Hello

**🔷 ৫. DOM দিয়ে Content পরিবর্তন**

javascript

CopyEdit

document.getElementById("demo").innerText = "New Text";

document.getElementById("demo").innerHTML = "<b>Bold Text</b>";

**🔷 ৬. DOM দিয়ে Style পরিবর্তন**

javascript

CopyEdit

const el = document.getElementById("demo");

el.style.color = "red";

el.style.fontSize = "20px";

**🔷 ৭. New Element তৈরি ও যোগ করা**

javascript

CopyEdit

const para = document.createElement("p");

para.innerText = "আমি নতুন প্যারাগ্রাফ।";

document.body.appendChild(para);

**🔷 ৮. Element মুছে ফেলা**

javascript

CopyEdit

const item = document.getElementById("demo");

item.remove(); // সরিয়ে ফেলে

**🔷 ৯. DOM Event Handling (ইভেন্টের মাধ্যমে কাজ)**

html

CopyEdit

<button onclick="sayHello()">Click Me</button>

javascript

CopyEdit

function sayHello() {

alert("হ্যালো!");

}

➡️ ইভেন্ট যেমন: click, mouseover, keydown, submit ইত্যাদি

**🔷 ১০. addEventListener()**

javascript

CopyEdit

const btn = document.getElementById("btn");

btn.addEventListener("click", function () {

alert("তুমি আমাকে ক্লিক করেছো!");

});

**🔷 ১১. Input Field থেকে Value নেওয়া**

html

CopyEdit

<input type="text" id="nameInput">

<button onclick="showName()">Show</button>

<p id="output"></p>

javascript

CopyEdit

function showName() {

const name = document.getElementById("nameInput").value;

document.getElementById("output").innerText = "Hello, " + name;

}

**✅ DOM নিয়ে গুরুত্বপূর্ণ জিনিস:**

| **বিষয়** | **উদাহরণ** |
| --- | --- |
| Element Access | document.getElementById("demo") |
| Text পড়া/বদলানো | innerText, innerHTML |
| Style পরিবর্তন | element.style.color = "blue" |
| নতুন element যোগ | createElement() + appendChild() |
| মুছে ফেলা | element.remove() |
| Event | onclick, addEventListener() |
| Input থেকে মান | element.value |

**🔚 উপসংহার:**

* DOM = JavaScript এর মাধ্যমে HTML Structure নিয়ন্ত্রণ
* DOM দিয়ে HTML পড়া, লেখা, style পরিবর্তন সব করা যায়
* DOM ছাড়া dynamic ওয়েবপেজ তৈরি সম্ভব নয়

**🏹 ১. Arrow Function – বাংলা নোট**

**🔷 Arrow Function কী?**

🔹 এটি হলো JavaScript-এ ফাংশন লেখার **ছোট এবং আধুনিক** পদ্ধতি (introduced in ES6)।  
🔹 এটি সাধারণ function keyword-এর বিকল্প।

**✅ Basic Syntax:**

javascript

CopyEdit

const functionName = (parameter) => {

// কাজ

};

**✅ উদাহরণ ১ – সাধারণ ফাংশনের পরিবর্তে:**

javascript

CopyEdit

// সাধারণ ফাংশন

function add(a, b) {

return a + b;

}

// Arrow function

const add = (a, b) => a + b;

console.log(add(3, 2)); // 5

**✅ উদাহরণ ২ – এক লাইনে Return:**

javascript

CopyEdit

const double = x => x \* 2;

console.log(double(5)); // 10

যদি একটাই parameter থাকে, তাহলে ( ) না দিলেও চলে।

**✅ উদাহরণ ৩ – কোনো parameter না থাকলে:**

javascript

CopyEdit

const greet = () => console.log("Hello!");

greet(); // Hello!

**✅ Arrow Function vs Regular Function (সারাংশ):**

| **পার্থক্য** | **Arrow Function** | **Regular Function** |
| --- | --- | --- |
| Syntax | ছোট | বড় |
| this binding | outer scope থেকে নেয় | নিজস্ব scope থাকে |
| constructor হিসেবে ব্যবহার | না | হ্যাঁ |
| arguments object | থাকে না | থাকে |

**🔥 গুরুত্বপূর্ণ: Arrow function এ this কিভাবে কাজ করে?**

javascript

CopyEdit

const person = {

name: "Rafia",

sayHi: () => {

console.log(`Hi, I am ${this.name}`);

}

};

person.sayHi(); // undefined (কারণ this কাজ করেনি)

➡️ Arrow function এ this **parent scope** থেকে আসে, তাই object context ধরে না।

**🔐 ২. Closure – বাংলা নোট**

**🔷 Closure কী?**

📌 **Closure** হলো এমন একটি ফাংশন যেটা **নিজের ভেতরের scope থেকে ভেরিয়েবল মনে রাখে**, এমনকি সেই scope-এর বাইরে থেকেও।

➡️ অন্যভাবে বললে:  
**"একটি inner function তার parent function এর variable access করতে পারে, এমনকি parent function শেষ হয়ে গেলেও।"**

**✅ Closure উদাহরণ:**

javascript

CopyEdit

function outer() {

let count = 0;

function inner() {

count++;

console.log(count);

}

return inner;

}

const counter = outer();

counter(); // 1

counter(); // 2

counter(); // 3

**🔍 ব্যাখ্যা:**

* inner() ফাংশন return হয়েছে কিন্তু এখনো count এর মান access করতে পারছে।
* কারণ, inner() ফাংশন **closure তৈরি করেছে** – যার ফলে outer scope এর ভেরিয়েবল ধরে রেখেছে।

**✅ বাস্তব জীবনের Closure ব্যবহার:**

**🔸 Button click counter:**

javascript

CopyEdit

function createCounter() {

let count = 0;

return function () {

count++;

return count;

};

}

const clickCounter = createCounter();

console.log(clickCounter()); // 1

console.log(clickCounter()); // 2

**✅ কেন Closure দরকার?**

| **দরকার** | **ব্যাখ্যা** |
| --- | --- |
| ✅ ডেটা গোপন রাখতে | ভেতরের ভেরিয়েবল বাইরে থেকে access করা যায় না |
| ✅ Private variable | Closure এর মাধ্যমে encapsulation করা যায় |
| ✅ Memory efficient | শুধু প্রয়োজনীয় scope ধরে রাখে |

**🧠 Arrow Function + Closure**

Arrow function দিয়েও closure তৈরি করা যায়:

javascript

CopyEdit

const outer = () => {

let value = 100;

return () => console.log(value);

};

const inner = outer();

inner(); // 100

**✅ Closure vs Normal Scope**

| **বিষয়** | **Scope** | **Closure** |
| --- | --- | --- |
| কতক্ষণ টিকে থাকে | Function শেষ হলে শেষ | Closure এর কারণে টিকে থাকে |
| Variable access | নিজের ভিতরের অংশ | বাইরে থেকেও access |

**🔚 উপসংহার:**

🔹 **Arrow Function** = ছোট, সুন্দর, this না বাঁধা modern ফাংশন  
🔹 **Closure** = একটি powerful concept যা function এর ভেতরের variable কে বাঁচিয়ে রাখে

**JavaScript এর Type (টাইপ) বিস্তারিত**

**১. JavaScript এ Type কী?**

* Type মানে হলো ভেরিয়েবল বা ডাটা যে ধরনের, যেমন সংখ্যা, লেখা, সত্য-মিথ্যা ইত্যাদি।
* JavaScript হলো **dynamically typed language**, অর্থাৎ variable declare করার সময় type নির্দিষ্ট করতে হয় না, বরং value based on type automatically ঠিক হয়।

**২. JavaScript এর Primitive Types (প্রিমিটিভ টাইপ)**

**২.১ Number (নাম্বার)**

* সংখ্যা যেমন ৫, ১০, ৩.১৪
* JavaScript সব সংখ্যাকে ফ্লোটিং পয়েন্ট হিসেবে ধরে।

js

CopyEdit

let num = 25;

let price = 99.99;

**২.২ String (স্ট্রিং)**

* লেখা বা অক্ষর সমষ্টি।
* ডাবল বা সিঙ্গেল কোটেশনে লেখা হয়।

js

CopyEdit

let name = "Rafia";

**২.৩ Boolean (বুলিয়ান)**

* true বা false মান।

js

CopyEdit

let isLoggedIn = true;

**২.৪ Undefined (আনডিফাইন্ড)**

* যখন ভেরিয়েবল ডিক্লেয়ার করা হয় কিন্তু কোনো মান দেওয়া হয় না।

js

CopyEdit

let x;

console.log(x); // undefined

**২.৫ Null (নাল)**

* ইচ্ছাকৃতভাবে কোনো মান খালি রেখে দেয়া হয়।

js

CopyEdit

let y = null;

**২.৬ Symbol (সিম্বল)**

* ইউনিক ও ইমিউটেবল টাইপ, সাধারণত অবজেক্টের property key হিসেবে ব্যবহার হয়।

js

CopyEdit

let sym = Symbol('id');

**২.৭ BigInt (বিগইনট)**

* বড় সংখ্যা ধারণের জন্য।

js

CopyEdit

let bigIntNum = 12345678901234567890n;

**৩. Non-Primitive Types (নন-প্রিমিটিভ টাইপ)**

**৩.১ Object (অবজেক্ট)**

* কী-ভ্যালু জোড়া ডাটা ধরে রাখে।

js

CopyEdit

let person = {

name: "Rafia",

age: 22

};

**৩.২ Array (অ্যারেএ)**

* ordered ডাটার collection, অবজেক্টের subtype।

js

CopyEdit

let arr = [1, 2, 3];

**৩.৩ Function (ফাংশন)**

* কোডের reusable ব্লক, যা অবজেক্ট টাইপেরই একটি বিশেষ রূপ।

js

CopyEdit

function greet() {

console.log("Hello");

}

**৪. JavaScript এ টাইপ কিভাবে চেক করবেন?**

* typeof অপারেটর দিয়ে কোনো ডাটার টাইপ পাওয়া যায়।

js

CopyEdit

console.log(typeof 25); // number

console.log(typeof "hello"); // string

console.log(typeof true); // boolean

console.log(typeof undefined); // undefined

console.log(typeof null); // object (এটি একটি JavaScript এর বাগ)

console.log(typeof {}); // object

console.log(typeof function(){});// function

console.log(typeof Symbol()); // symbol

console.log(typeof 10n); // bigint

**৫. JavaScript এর টাইপ সম্পর্কিত গুরুত্বপূর্ণ বিষয়সমূহ**

* **Dynamic Typing**: ভেরিয়েবলের টাইপ পরিবর্তন হতে পারে যখন নতুন ভ্যালু অ্যাসাইন করা হয়।

js

CopyEdit

let data = 5; // number

data = "hello"; // এখন string

* **Type Coercion**: JavaScript প্রয়োজনে স্বয়ংক্রিয়ভাবে টাইপ কনভার্সন করে।

js

CopyEdit

console.log(5 + "5"); // "55" (number + string = string concatenation)

console.log("5" \* 2); // 10 (string \* number = number multiplication)

* **Strict Equality (===) vs Loose Equality (==)**:

js

CopyEdit

5 == "5" // true (value only check করে)

5 === "5" // false (value এবং type উভয় চেক করে)

**৬. Summary**

| **টাইপ** | **উদাহরণ** | **বর্ণনা** |
| --- | --- | --- |
| Number | 10, 3.14 | সংখ্যা |
| String | "Hello", 'Rafia' | লেখা বা স্ট্রিং |
| Boolean | true, false | সত্য-মিথ্যা |
| Undefined | undefined | মান না থাকা |
| Null | null | মান intentionally খালি রাখা |
| Symbol | Symbol('id') | ইউনিক ও ইমিউটেবল |
| BigInt | 1234567890123n | বড় সংখ্যা |
| Object | {name: "Rafia"} | কী-ভ্যালু ডাটা |
| Function | function() {...} | ফাংশন (কোড ব্লক) |

**JavaScript এ Value (ভ্যালু) - বিস্তারিত**

**১. Value কি?**

* Value হলো যেটা কোনো ভেরিয়েবল বা এক্সপ্রেশন ধারণ করে বা প্রকাশ করে।
* সহজ ভাষায়, ভ্যালু হলো ডাটা বা মান, যা ব্যবহার করা হয় বা সংরক্ষণ করা হয়।

**২. JavaScript এ Value এর ধরন**

JavaScript এ বিভিন্ন ধরনের value থাকতে পারে, যেগুলো মূলত **data types** এর অন্তর্ভুক্ত।

উদাহরণস্বরূপ:

js

CopyEdit

let num = ১০; // Number value

let name = "রাফিয়া"; // String value

let isActive = true; // Boolean value

let person = { // Object value

name: "রাফিয়া",

age: ২২

};

**৩. Primitive Value (প্রিমিটিভ ভ্যালু)**

* সরাসরি মান, যেমন:

| **Type** | **Example Value** |
| --- | --- |
| Number | ১০, ৩.১৪ |
| String | "হ্যালো" |
| Boolean | true, false |
| Undefined | undefined |
| Null | null |
| Symbol | Symbol('id') |
| BigInt | 1234567890n |

**৪. Non-Primitive Value (নন-প্রিমিটিভ ভ্যালু)**

* অবজেক্ট বা অ্যারে বা ফাংশন, যেগুলো ভ্যালু হিসেবে জটিল ডাটা ধারণ করে।

js

CopyEdit

let arr = [1, 2, 3]; // Array value

let obj = {a: 10}; // Object value

function greet() { // Function value

console.log("Hello");

}

**৫. Value এবং Variable এর সম্পর্ক**

* Variable হলো একটা container, আর value হলো সেই container এর ভিতরে রাখা ডাটা।

js

CopyEdit

let x = ৫; // ৫ হলো x এর value

let y = "abc"; // "abc" হলো y এর value

**৬. Value এর পরিবর্তন (Changing Value)**

* Primitive value গুলো immutable অর্থাৎ সরাসরি পরিবর্তন হয় না, তবে ভেরিয়েবলের value পরিবর্তন করা যায়।

js

CopyEdit

let a = ১০;

a = ২০; // a এখন ২০ হলো

* Non-primitive value গুলো mutable, মানে অবজেক্ট বা অ্যারের ভিতরের data পরিবর্তন করা যায়।

js

CopyEdit

let person = {name: "রাফিয়া"};

person.name = "সাদিয়া"; // অবজেক্টের প্রপার্টি পরিবর্তন

**৭. Value vs Reference**

* Primitive value সরাসরি ভেরিয়েবলে রাখা হয় (value type)।
* Object এবং Array এর ক্ষেত্রে ভেরিয়েবল ডাটার reference (ঠিকানা) ধরে রাখে, ডাটা নিজেই নয়।

js

CopyEdit

let obj1 = {a: ১০};

let obj2 = obj1;

obj2.a = ২০;

console.log(obj1.a); // ২০ (কারণ একই রেফারেন্স)

**৮. Expression এ Value**

* JavaScript এর এক্সপ্রেশন evaluate করে একটা value produce করে।

js

CopyEdit

let sum = ৫ + ১০; // ১৫ হলো value

let str = "Hi " + "there"; // "Hi there" হলো value

**৯. Summary**

| **বিষয়** | **বর্ণনা** |
| --- | --- |
| Value | ভেরিয়েবল বা এক্সপ্রেশনের ধারণ করা মান |
| Primitive Value | সরাসরি ডাটা যেমন সংখ্যা, স্ট্রিং ইত্যাদি |
| Non-Primitive Value | অবজেক্ট, অ্যারে, ফাংশন ইত্যাদি |
| Value পরিবর্তন | Primitive immutable, Object mutable |
| Reference | অবজেক্ট ভেরিয়েবল রেফারেন্স ধারণ করে |
| **JavaScript এ Variable (ভেরিয়েবল) - বিস্তারিত**  **১. Variable কি?**   * Variable হলো একটা container বা জায়গা যেটাতে ডাটা বা value রাখা হয়। * সহজভাবে বললে, Variable হল একটা নাম যেটার মাধ্যমে আমরা ডাটা সংরক্ষণ ও ব্যবহার করি।   **২. JavaScript এ Variable ডিক্লেয়ার করার উপায়**  JavaScript এ Variable ডিক্লেয়ার করার জন্য তিনটি কীওয়ার্ড ব্যবহার হয়:   * var * let * const   **৩. var**   * পুরনো ধরনের variable ডিক্লেয়ার করার পদ্ধতি। * Function scope থাকে, অর্থাৎ শুধুমাত্র function এর ভিতরে কাজ করে। * Redeclare (আবার একই নামের variable ডিক্লেয়ার) করা যায়। * Value পরিবর্তন করা যায়।   js  CopyEdit  var name = "Rafia";  var name = "Sadia"; // redeclare allowed  name = "Nabila"; // update allowed  **৪. let**   * ES6 থেকে এসেছে। * Block scoped, অর্থাৎ {} এর মধ্যে সীমাবদ্ধ। * Redeclare করা যায় না, তবে value পরিবর্তন করা যায়।   js  CopyEdit  let age = 25;  age = 30; // update allowed  // let age = 35; // error: redeclare not allowed  **৫. const**   * Constant variable, অর্থাৎ একবার value দিলে পরে পরিবর্তন করা যায় না। * Block scoped। * Declare করার সময় value assign করা আবশ্যক।   js  CopyEdit  const PI = 3.14;  // PI = 3.1416; // error: value change not allowed  **৬. Variable এর Naming Rules (নামকরণ নিয়ম)**   * Variable নাম ইংরেজি অক্ষর, সংখ্যা, $ এবং \_ ব্যবহার করে লিখতে পারো। * Variable নাম সংখ্যা দিয়ে শুরু করতে পারবে না। * Variable নামের মধ্যে স্পেস দেওয়া যাবে না। * JavaScript এর reserved keywords (যেমন var, let, if, else) variable নাম হিসেবে ব্যবহার করা যাবে না।   **সঠিক উদাহরণ:**  js  CopyEdit  let userName;  let $amount;  let \_price;  let age25;  **ভুল উদাহরণ:**  js  CopyEdit  let 25age; // ভুল, সংখ্যা দিয়ে শুরু  let user name; // ভুল, স্পেস দেওয়া যাবে না  let var; // ভুল, reserved keyword  **৭. Variable এ Value Assign ও Update করা**  js  CopyEdit  let city = "Dhaka"; // value assign  city = "Chittagong"; // value update allowed (let, var)  const এর ক্ষেত্রে:  js  CopyEdit  const country = "Bangladesh";  // country = "India"; // error: update not allowed  **৮. Variable এর Scope (স্কোপ)**   * **Global scope:** স্কোপ যা পুরো প্রোগ্রামের জন্য খোলা থাকে। * **Function scope:** শুধুমাত্র function এর মধ্যে সীমাবদ্ধ (var এর ক্ষেত্রে)। * **Block scope:** শুধু {} এর মধ্যে সীমাবদ্ধ (let, const এর ক্ষেত্রে)।   js  CopyEdit  if (true) {  var x = 10; // global scoped (function scoped)  let y = 20; // block scoped  const z = 30; // block scoped  }  console.log(x); // 10 (accessible)  console.log(y); // error (not accessible)  console.log(z); // error (not accessible)  **৯. Hoisting (হোইস্টিং)**   * JavaScript এ var declared variable গুলো উপরে উঠিয়ে নেয় (hoist করে), কিন্তু let ও const এর ক্ষেত্রে হয় না। * এর মানে, var এর জন্য variable declare না করে আগে ব্যবহার করলে undefined দেখায়, কিন্তু let ও const এর জন্য error দেয়।   js  CopyEdit  console.log(a); // undefined  var a = 5;  console.log(b); // ReferenceError  let b = 10;  **১০. Summary টেবিল**   | **Keyword** | **Scope** | **Redeclare Allowed** | **Update Allowed** | **Initialize Required** | | --- | --- | --- | --- | --- | | var | Function | Yes | Yes | No | | let | Block | No | Yes | No | | const | Block | No | No | Yes |   **১১. উদাহরণ**  js  CopyEdit  var name = "Rafia";  let age = 22;  const PI = 3.14;  age = 23; // allowed  // PI = 3.1416; // error  **১২. Variable Best Practices**   * const ব্যবহার করো যতটা সম্ভব, কারন এটা নিরাপদ। * যখন value পরিবর্তন দরকার হয় তখন let ব্যবহার করো। * var এড়ানো ভালো, কারণ এর scope বুঝতে ভুল হতে পারে। |  |