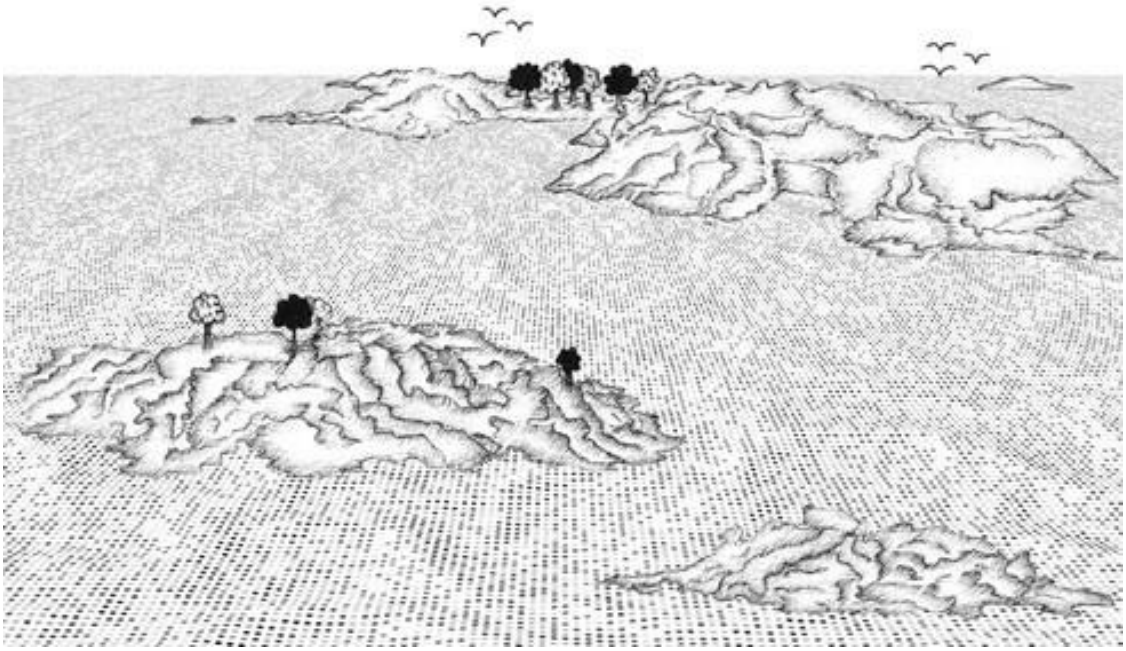
Table of Contents

অধ্যায় ১: মান (Values), ধরন (Types), এবং অপারেটর (Operators)



বিট এবং বাইনারি প্রতিনিধিত্বের চিত্র

> "যন্ত্রের পৃষ্ঠের নীচে, প্রোগ্রাম চলমান। বিনা প্রচেষ্টায়, এটি প্রসারিত ও সংকুচিত হয়। মহান সামঞ্জস্যে, ইলেকট্রনগুলি ছড়িয়ে পড়ে এবং পুনরায় একত্রিত হয়। মনিটরে দৃশ্যমান রূপগুলি জলের তরঙ্গের মতো। এর সারাংশ অদৃশ্যভাবে নীচে থাকে।"
>
> —মাস্টার ইউয়ান-মা, দ্য বুক অফ প্রোগ্রামিং

কম্পিউটারের জগতে, শুধুমাত্র ডাটা (data) রয়েছে। আপনি ডাটা পড়তে পারেন, পরিবর্তন করতে পারেন, নতুন ডাটা তৈরি করতে পারেন—কিন্তু যা ডাটা নয় তা উল্লেখ করা যায় না। সমস্ত ডাটা বিটের (bit) দীর্ঘ ক্রমের আকারে সংরক্ষিত থাকে এবং মৌলিকভাবে একই রকম।

সংখ্যা (Numbers)

জাভাস্ক্রিপ্ট (JavaScript) এ সংখ্যা টাইপের মান লেখা হয় এভাবে:

```
13
```

জাভাস্ক্রিপ্ট একটি নির্দিষ্ট সংখ্যক বিট (64 বিট) ব্যবহার করে একটি সংখ্যা মান সংরক্ষণ করে। 64 বিট দিয়ে আপনি 2^{64} টি ভিন্ন সংখ্যা প্রকাশ করতে পারেন।

ভগ্নাংশ সংখ্যা লেখা হয় দশমিক বিন্দু (decimal point) ব্যবহার করে:

```
9.81
```

খুব বড় বা খুব ছোট সংখ্যার জন্য, আপনি বৈজ্ঞানিক সংকেতন (scientific notation) ব্যবহার করতে পারেন *e* (exponent) যোগ করে:

```
2.998e8  // 299,800,000 (আলোর গতি মিটার/সেকেন্ডে)
```

স্ট্রিং (Strings)

টেক্সট ডাটা স্ট্রিং হিসেবে প্রকাশ করা হয়। একক উদ্ধৃতি (single quotes) (''), দ্বৈত উদ্ধৃতি (double quotes) ("") অথবা ব্যাকটিক (backticks) (`) ব্যবহার করে স্ট্রিং লেখা যায়:

```
`সমুদ্রের নীচে`
"মহাসাগরে শায়িত"
'মহাসাগরে ভাসমান'
```

টেমপ্লেট লিটারাল (template literals) (`) ব্যবহার করে আপনি বহুলাইন স্ট্রিং এবং ডায়নামিক মান অন্তর্ভুক্ত করতে পারেন:

```
`100 এর অর্ধেক হল ${100 / 2}`  // ফলাফল: "100 এর অর্ধেক হল 50"
```

বুলিয়ান (Boolean)

বুলিয়ান টাইপের মান শুধুমাত্র দুটি: `true` এবং `false`। এগুলি "হ্যাঁ" এবং "না" বা "চালু" (on) এবং "বন্ধ" (off) অবস্থা প্রকাশ করতে ব্যবহৃত হয়।

তুলনামূলক অপারেটর (comparison operators): - > (বড়/greater than) - < (ছোট/less than) - >= (বড় বা সমান/greater than or equal to) - <= (ছোট বা সমান/less than or equal to) - == (সমান/equal to) - === (সঠিক সমান/strictly equal to) - != (অসমান/not equal to) - !== (সঠিক অসমান/strictly not equal to)

যুক্তি অপারেটর (logical operators): - && (এবং/and) - || (অথবা/or) - ! (না/not)

খালি মান (Empty Values)

`null` এবং `undefined` দুটি বিশেষ মান যা অর্থপূর্ণ মানের অনুপস্থিতি (absence of a meaningful value) নির্দেশ করে।

স্বয়ংক্রিয় টাইপ রূপান্তর (Type Coercion)

জাভাস্ক্রিপ্ট স্বয়ংক্রিয়ভাবে ভিন্ন টাইপের মানগুলিকে রূপান্তর করতে পারে:

```
"5" - 1    // ফলাফল: 4
"5" + 1    // ফলাফল: "51"
8 * null   // ফলাফল: 0
```

`??` অপারেটর (nullish coalescing operator): এটি `||` অপারেটরের মতো কাজ করে, তবে কেবল বাম দিকের মান `null` বা `undefined` হলেই ডান দিকের মান ফেরত দেয়:

```
0 ?? 100      // ফলাফল: 0
null ?? 100   // ফলাফল: 100
```

অধ্যায় ২: প্রোগ্রাম কাঠামো (Program Structure)

> “আর আমার হৃদয় উজ্জ্বল লাল হয়ে ওঠে আমার স্বচ্ছ, পাতলা ত্বকের নীচে এবং তারা আমাকে ফিরিয়ে আনার জন্য 10cc JavaScript প্রয়োগ করতে হয়। (আমি রক্তে বিষাক্ত পদার্থে ভালো সাড়া দিই।) মানুষ, এই জিনিসটি আপনার গিলস থেকে পীচগুলোকে বের করে দেবে!”
>
> —_why, Why’s (Poignant) Guide to Ruby

এই অধ্যায়ে, আমরা এমন কিছু করতে শুরু করব যাকে সত্যিকারের *প্রোগ্রামিং (programming)* বলা যায়। আমরা JavaScript ভাষার উপর আমাদের নিয়ন্ত্রণকে বাড়িয়ে নিয়ে যাব - শুধু বিচ্ছিন্ন শব্দ ও বাক্যাংশের বাইরে, যেখানে আমরা অর্থপূর্ণ গদ্য প্রকাশ করতে পারব।

এক্সপ্রেশন এবং স্টেটমেন্ট (Expressions and Statements)

কোডের একটি অংশ যা একটি মান উৎপন্ন করে তাকে *এক্সপ্রেশন (expression)* বলে। প্রতিটি মান যা সরাসরি লেখা হয় (যেমন 22 বা "মনোবিশ্লেষণ") তা একটি এক্সপ্রেশন।

যদি একটি এক্সপ্রেশন বাক্যের একটি অংশের সাথে সামঞ্জস্যপূর্ণ হয়, তবে একটি JavaScript *স্টেটমেন্ট (statement)* সম্পূর্ণ বাক্যের সাথে সামঞ্জস্যপূর্ণ। একটি প্রোগ্রাম হল স্টেটমেন্টের একটি তালিকা।

বাইন্ডিং (Bindings)

একটি প্রোগ্রাম কীভাবে অভ্যন্তরীণ অবস্থা বজায় রাখে? এটি কীভাবে জিনিস মনে রাখে? `let` কীওয়ার্ড (keyword) একটি নতুন বাইন্ডিং সংজ্ঞায়িত করে:

```
let caught = 5 * 5;
```

নিয়ন্ত্রণ প্রবাহ (Control Flow)

যখন আপনার প্রোগ্রামে একাধিক স্টেটমেন্ট থাকে, তখন স্টেটমেন্টগুলি উপর থেকে নীচে পর্যন্ত কার্যকর হয়।

শর্তসাপেক্ষ কার্যকরণ (Conditional Execution)

সব প্রোগ্রামই সরল রাস্তা নয়। কখনও কখনও আমরা একটি শাখাযুক্ত পথ তৈরি করতে চাই যেখানে প্রোগ্রাম পরিস্থিতি অনুযায়ী সঠিক শাখা বেছে নেয়। এটিকে *শর্তসাপেক্ষ কার্যকরণ* বলে:

```
let theNumber = Number(prompt("একটি সংখ্যা বেছে নিন"));
if (!Number.isNaN(theNumber)) {
  console.log("আপনার সংখ্যাটি হল " + theNumber * theNumber + " এর বর্গমূল");
} else {
  console.log("হে! আপনি আমাকে একটি সংখ্যা দেননি কেন?");
}
```

লুপ (Loops)

কল্পনা করুন এমন একটি প্রোগ্রাম যা 0 থেকে 12 পর্যন্ত সব জোড় সংখ্যা আউটপুট করে। এটি লেখার একটি উপায়:

```
let number = 0;
while (number <= 12) {
  console.log(number);
  number = number + 2;
}
```

একটি `while` লুপ তখনই চলতে থাকে যতক্ষণ এর শর্ত সত্য থাকে।

`for` লুপ (for loop) এর ব্যবহার:

```
for (let number = 0; number <= 12; number += 2) {
  console.log(number);
}
```

লুপ থেকে বের হওয়া (Breaking Out of a Loop)

কখনও কখনও আপনাকে একটি লুপ থেকে মাঝপথে বের হতে হতে পারে। এজন্য JavaScript-এ `break` স্টেটমেন্ট (break statement) রয়েছে।

```
for (let current = 20; ; current = current + 1) {
  if (current % 7 == 0) {
    console.log(current);
    break;
  }
}
// → 21
```

`continue` স্টেটমেন্ট (continue statement) `break` এর মতোই, তবে এটি লুপ থেকে বের না হয়ে পরবর্তী পুনরাবৃত্তিতে চলে যায়।

বাইন্ডিং আপডেট করা (Updating Bindings)

লুপের ক্ষেত্রে, প্রায়শই একটি বাইন্ডিংকে তার আগের মানের উপর ভিত্তি করে আপডেট করতে হয়:

```
counter = counter + 1;
```

JavaScript এর জন্য শর্টকাট (shortcuts) প্রদান করে:

```
counter += 1;
```

কমেন্ট (Comments)

JavaScript-এ দুই ধরনের কমেন্ট লেখা যায়:

```
// এটি একটি সিঙ্গেল-লাইন কমেন্ট (single-line comment)

/* এটি একটি মাল্টি-লাইন কমেন্ট (multi-line comment)
   যা একাধিক লাইন জুড়ে থাকতে পারে */
```

অনুশীলনী (Exercises)

ত্রিভুজ লুপ (Triangle Loop)

এমন একটি লুপ লিখুন যা `console.log` ব্যবহার করে নিচের ত্রিভুজটি আঁকে:

```
#
##
###
####
#####
######
#######
```

ফিজবাজ (FizzBuzz)

এমন একটি প্রোগ্রাম লিখুন যা 1 থেকে 100 পর্যন্ত সংখ্যাগুলি প্রিন্ট করে, তবে দুটি ব্যতিক্রম সহ: - 3 দ্বারা বিভাজ্য সংখ্যাগুলির জন্য, সংখ্যার পরিবর্তে `"Fizz"` প্রিন্ট করুন - 5 দ্বারা বিভাজ্য সংখ্যাগুলির জন্য (কিন্তু 3 দ্বারা নয়), `"Buzz"` প্রিন্ট করুন - যদি সংখ্যাটি 3 এবং 5 উভয় দ্বারা বিভাজ্য হয়, তবে `"FizzBuzz"` প্রিন্ট করুন

চেসবোর্ড (Chess Board)

এমন একটি প্রোগ্রাম লিখুন যা 8×8 গ্রিড (grid) তৈরি করে, যেখানে প্রতিটি ঘরে স্পেস অথবা “#” চিহ্ন থাকবে, যাতে চেসবোর্ডের মতো দেখায়:

```
 # # # #
# # # #
 # # # #
# # # #
 # # # #
# # # #
 # # # #
# # # #
```

অধ্যায় ৩: ফাংশন (Functions)

> “মানুষ মনে করে কম্পিউটার বিজ্ঞান প্রতিভাবানদের শিল্প, কিন্তু বাস্তবতা এর ঠিক উল্টো - এটি অনেক মানুষের কাজের সমন্বয়, যেখানে প্রত্যেকে ছোট ছোট পাথরের মতো একটি দেয়াল গড়ে তোলে।”
>
> —ডোনাল্ড কনুথ (Donald Knuth)

ফাংশন (functions) হল JavaScript প্রোগ্রামিংয়ের সবচেয়ে গুরুত্বপূর্ণ বিল্ডিং ব্লক (building blocks)। প্রোগ্রামের একটি অংশকে একটি মানে মোড়ানোর ধারণাটির অনেক ব্যবহার রয়েছে। এটি আমাদেরকে বড় প্রোগ্রামগুলিকে কাঠামোবদ্ধ করার, পুনরাবৃত্তি কমানোর, সাব-প্রোগ্রামগুলির সাথে নাম যুক্ত করার এবং এই সাব-প্রোগ্রামগুলিকে একে অপরের থেকে বিচ্ছিন্ন করার উপায় দেয়।

যেভাবে ফাংশন কাজ করে (How Functions Work)

ফাংশন নিজস্ব স্কোপ (scope) তৈরি করে। একটি ফাংশন প্যারামিটার (parameters) গ্রহণ করতে পারে এবং একটি মান রিটার্ন (return) করতে পারে:

```
function square(x) {
  return x * x;
}

console.log(square(4));  // → 16
console.log(square(5));  // → 25
```

আর্গুমেন্ট অবজেক্ট (The Arguments Object)

প্রতিটি ফাংশন কলে একটি বিশেষ ভেরিয়েবল (variable) `arguments` থাকে। এটি একটি অবজেক্ট যা ফাংশনে পাঠানো সব আর্গুমেন্ট (arguments) ধারণ করে:

```
function showArgs() {
  return arguments;
}
console.log(showArgs("a", "b"));  // → {"0": "a", "1": "b"}
```

ফাংশন এক্সপ্রেশন (Function Expression)

ফাংশন একটি মান হিসেবে প্রকাশ করা যায়। এই ধরনের ফাংশনকে *ফাংশন এক্সপ্রেশন (function expression)* বলে:

```
const square = function(x) {
  return x * x;
};
```

অ্যারো ফাংশন (Arrow Functions)

ফাংশন লেখার একটি সংক্ষিপ্ত উপায় হল অ্যারো ফাংশন (arrow functions):

```
const square = x => x * x;
const add = (a, b) => a + b;
```

অনুশীলনী (Exercises)

ফ্যাক্টরিয়াল (Factorial)

একটি সংখ্যার ফ্যাক্টরিয়াল (factorial) গণনা করে এমন একটি রিকার্সিভ (recursive) ফাংশন লিখুন।

ফিবোনাচ্চি (Fibonacci)

ফিবোনাচ্চি সিরিজের (Fibonacci series) n-তম সংখ্যা বের করার জন্য একটি ফাংশন লিখুন।

অধ্যায় ৪: ডাটা স্ট্রাকচার (Data Structures): অবজেক্ট (Objects) এবং অ্যারে (Arrays)

> “একটি প্রোগ্রামের কাঠামো ডাটার কাঠামোকে প্রতিফলিত করে। জটিল প্রোগ্রামগুলি সরল ডাটা স্ট্রাকচার থেকে আসে, আর জটিল ডাটা স্ট্রাকচার থেকে আসে সরল প্রোগ্রামিং।”
>
> —চার্লস ব্যাবেজ (Charles Babbage), Passages from the Life of a Philosopher (1864)

এই অধ্যায়ে, আমরা সংযোজিত ডাটা স্ট্রাকচার (compound data structures) নিয়ে আলোচনা করব। অবজেক্ট (objects) আপনাকে মানের সংগ্রহ একসাথে গ্রুপ করতে দেয় এবং এই মানগুলির মধ্যে জটিল সম্পর্ক প্রকাশ করতে দেয়।

দ্য ওয়েরস্কুইরেল (The Weresquirrel)

জ্যাক্স একজন প্রোগ্রামার। প্রতি সপ্তাহে কয়েকবার, সূর্যাস্তের পরে, তিনি একটি কাঠবিড়ালিতে পরিণত হন। এই অদ্ভুত অবস্থার কারণ খুঁজে বের করার জন্য, তিনি একটি লগ (log) রাখা শুরু করেন - প্রতিদিন কী কী কাজ করেছেন এবং কাঠবিড়ালিতে রূপান্তরিত হয়েছিলেন কি না।

অ্যারে (Arrays)

অ্যারে (arrays) হল একই ধরনের কয়েকটি মানের সংগ্রহ। মানগুলি ক্রমিক সংখ্যা দিয়ে সূচিবদ্ধ করা হয়:

```
let listOfNumbers = [2, 3, 5, 7, 11];
console.log(listOfNumbers[0]);
// → 2
console.log(listOfNumbers[2]);
// → 5
```

প্রপার্টি (Properties)

প্রায় সব JavaScript মানেরই প্রপার্টি (properties) থাকে। একটি প্রপার্টি অ্যাক্সেস করার দুটি মূল উপায় আছে: - ডট নোটেশন (dot notation): `value.x` - ব্র্যাকেট নোটেশন (bracket notation): `value[x]`

```
let array = [1, 2, 3];
console.log(array.length);
// → 3
console.log(array["length"]);
// → 3
```

মেথড (Methods)

স্ট্রিং এবং অ্যারে উভয়েরই এমন কিছু প্রপার্টি থাকে যা ফাংশন মান ধারণ করে। এই ধরনের প্রপার্টিগুলিকে *মেথড (methods)* বলা হয়:

```
let doh = "Doh";
console.log(typeof doh.toUpperCase);
// → function
```

```
console.log(doh.toUpperCase());
// → DOH
```

অ্যারে মেথড (Array Methods)

অ্যারেতে কিছু বিল্ট-ইন মেথড (built-in methods) রয়েছে:

```
let sequence = [1, 2, 3];
sequence.push(4);
sequence.push(5);
console.log(sequence);
// → [1, 2, 3, 4, 5]
console.log(sequence.pop());
// → 5
console.log(sequence);
// → [1, 2, 3, 4]
```

REST প্যারামিটার (REST Parameters)

একটি ফাংশন যেকোনো সংখ্যক আর্গুমেন্ট (arguments) নিতে পারে। এটি করার জন্য, প্যারামিটারের আগে তিনটি ডট (...) ব্যবহার করুন:

```
function max(...numbers) {
  let result = -Infinity;
  for (let number of numbers) {
    if (number > result) result = number;
  }
  return result;
}
console.log(max(4, 1, 9, -2));
// → 9
```

JSON (JavaScript Object Notation)

জাভাস্ক্রিপ্ট মানগুলি টেক্সট হিসাবে সংরক্ষণ করা যায় JSON ফরম্যাট (format) ব্যবহার করে:

```
let string = JSON.stringify({squirrel: false,
                             events: ["weekend"]});
console.log(string);
// → {"squirrel":false,"events":["weekend"]}

// JSON থেকে JavaScript অবজেক্টে রূপান্তর (convert)
console.log(JSON.parse(string).events);
// → ["weekend"]
```

অবজেক্ট (Objects)

অ্যারে একটি বিশেষ ধরনের অবজেক্ট যা ক্রমিক সংখ্যা দ্বারা সূচিবদ্ধ করা হয়। সাধারণ অবজেক্টে যে কোন নাম দিয়ে প্রপার্টি থাকতে পারে:

```
let day1 = {
  squirrel: false,
  events: ["কাজ", "গাছ স্পর্শ করা", "পিজ্জা খাওয়া"]
};
console.log(day1.squirrel);
// → false
console.log(day1.wolf);
// → undefined
```

অনুশীলনী (Exercises)

দ্য সাম অফ এ রেঞ্জ (The Sum of a Range)

`range` এবং `sum` নামে দুটি ফাংশন লিখুন। প্রথমটি দুটি আর্গুমেন্ট, `start` এবং `end` নেয় এবং এই সীমার মধ্যে সব সংখ্যা সহ একটি অ্যারে রিটার্ন করে। দ্বিতীয়টি একটি সংখ্যার অ্যারে নেয় এবং এর সব উপাদানের যোগফল রিটার্ন করে।

রিভার্সিং এন অ্যারে (Reversing an Array)

দুটি ফাংশন লিখুন, `reverseArray` এবং `reverseArrayInPlace`। প্রথমটি একটি অ্যারে নেয় এবং একই উপাদান নিয়ে নতুন একটি অ্যারে তৈরি করে উল্টো ক্রমে। দ্বিতীয়টি একই কাজ করে তবে দেওয়া অ্যারেটিকেই পরিবর্তন করে।

অধ্যায় ৫: হায়ার-অর্ডার ফাংশন (Higher-Order Functions)

> “ট্রিকস অফ দ্য ট্রেড প্রোগ্রামিং নয়...ট্রেড অফ দ্য ট্রিকস প্রোগ্রামিং।”
>
> —সি.এ.আর. হোর (C.A.R. Hoare), 1980 ACM টিউরিং অ্যাওয়ার্ড লেকচার

একটি প্রোগ্রামকে অ্যাবস্ট্রাক্ট (abstract) করার অর্থ হল এর বিস্তারিত বিবরণগুলিকে সরল করা যাতে আমরা বড় চিত্রটি দেখতে পাই। একটি অ্যাবস্ট্রাকশন (abstraction) একটি জটিল কাজকে সহজ শব্দে প্রকাশ করে।

অ্যাবস্ট্রাকশন (Abstraction)

অ্যাবস্ট্রাকশন প্রোগ্রামকে সরল করার একটি উপায়। নিচের উদাহরণে, আমরা একটি জটিল অপারেশনকে একটি সহজ ফাংশনে মোড়ানো দেখতে পাব:

```
function repeat(n, action) {
  for (let i = 0; i < n; i++) {
    action(i);
  }
}

repeat(3, console.log);
// → 0
// → 1
// → 2
```

হায়ার-অর্ডার ফাংশন (Higher-Order Functions)

ফাংশনগুলি যে অন্য ফাংশনগুলিকে আর্গুমেন্ট হিসেবে নেয় বা ফাংশন রিটার্ন করে, তাদেরকে *হায়ার-অর্ডার ফাংশন* বলে। এরা আমাদেরকে অ্যাবস্ট্রাকশনের নতুন ধরণ তৈরি করতে দেয়:

```
function greaterThan(n) {
  return m => m > n;
}
let greaterThan10 = greaterThan(10);
console.log(greaterThan10(11));
// → true
```

অ্যারে মেথড (Array Methods)

JavaScript-এ অনেকগুলি বিল্ট-ইন হায়ার-অর্ডার অ্যারে মেথড রয়েছে:

forEach (ForEach)

```
["A", "B"].forEach(l => console.log(l));
// → A
// → B
```

filter (Filter)

```
console.log([1, 2, 3, 4, 5].filter(n => n % 2 == 0));
// → [2, 4]
```

map (Map)

```
console.log([1, 2, 3].map(n => n * 2));
// → [2, 4, 6]
```

reduce (Reduce)

```
console.log([1, 2, 3, 4].reduce((a, b) => a + b));
// → 10
```

অধ্যায় ৬: অবজেক্টের গোপন জীবন (The Secret Life of Objects)

> “একটি অবজেক্ট হল একটি ব্যাগ যার মধ্যে প্রপার্টি রাখা যায়।”
>
> —বারবারা লিসকভ (Barbara Liskov), Programming with Abstract Data Types

অ্যাবস্ট্রাক্ট ডাটা টাইপ (Abstract Data Types)

অবজেক্ট-ওরিয়েন্টেড প্রোগ্রামিং (Object-Oriented Programming) এর মূল ধারণা হল ডাটা (data) এবং ফাংশনালিটি (functionality) একসাথে প্যাকেজ করা। এটি আমাদের কোডকে আরও ভাল সংগঠিত করতে সাহায্য করে।

মেথড (Methods)

JavaScript-এ, মেথড হল এমন প্রপার্টি যা ফাংশন মান ধারণ করে। উদাহরণ:

```
function speak(line) {
  console.log(`The ${this.type} rabbit says '${line}'`);
}
let whiteRabbit = {type: "white", speak};
let hungryRabbit = {type: "hungry", speak};

whiteRabbit.speak("Oh my fur and whiskers");
// → The white rabbit says 'Oh my fur and whiskers'
```

প্রোটোটাইপ (Prototypes)

JavaScript-এ প্রোটোটাইপ (prototypes) হল অবজেক্টের মধ্যে প্রপার্টি শেয়ার করার একটি মেকানিজম:

```
let protoRabbit = {
  speak(line) {
    console.log(`The ${this.type} rabbit says '${line}'`);
  }
};
let killerRabbit = Object.create(protoRabbit);
killerRabbit.type = "killer";
killerRabbit.speak("SKREEEE!");
// → The killer rabbit says 'SKREEEE!'
```

ক্লাস (Classes)

একটি ক্লাস (class) একটি অবজেক্টের ব্লুপ্রিন্ট (blueprint) নির্ধারণ করে:

```
class Rabbit {
  constructor(type) {
    this.type = type;
  }
  speak(line) {
```

```
    console.log(`The ${this.type} rabbit says '${line}'`);
  }
}

let killerRabbit = new Rabbit("killer");
let blackRabbit = new Rabbit("black");
```

প্রাইভেট প্রপার্টি (Private Properties)

JavaScript-এ প্রাইভেট প্রপার্টি (private properties) এর সামনে # চিহ্ন থাকে:

```
class Temperature {
  #celsius;
  constructor(celsius) {
    this.#celsius = celsius;
  }
  get fahrenheit() {
    return this.#celsius * 1.8 + 32;
  }
  set fahrenheit(value) {
    this.#celsius = (value - 32) / 1.8;
  }
}
```

উত্তরাধিকার (Inheritance)

একটি ক্লাস অন্য একটি ক্লাসের বৈশিষ্ট্য এবং মেথড পেতে পারে। এটিকে *উত্তরাধিকার (inheritance)* বলে:

```
class Matrix {
  constructor(width, height, element = (x, y) => undefined) {
    this.width = width;
    this.height = height;
    this.content = [];

    for (let y = 0; y < height; y++) {
      for (let x = 0; x < width; x++) {
        this.content[y * width + x] = element(x, y);
      }
    }
  }
}

class SymmetricMatrix extends Matrix {
  constructor(size, element = (x, y) => undefined) {
    super(size, size, (x, y) => {
      if (x < y) return element(y, x);
      else return element(x, y);
    });
```

```
  }
}
```

instanceof অপারেটর (Operator)

`instanceof` অপারেটর একটি অবজেক্ট কোন ক্লাসের ইনস্ট্যান্স (instance) কিনা তা পরীক্ষা করে:

```
console.log(new SymmetricMatrix(2) instanceof SymmetricMatrix);
// → true
console.log(new SymmetricMatrix(2) instanceof Matrix);
// → true
console.log(new Matrix(2, 2) instanceof SymmetricMatrix);
// → false
```

অনুশীলনী (Exercises)

ভেক্টর টাইপ (Vector Type)

দ্বি-মাত্রিক স্থানে একটি ভেক্টর প্রকাশ করার জন্য `Vec` নামে একটি ক্লাস লিখুন। এতে দুটি প্রপার্টি থাকবে, `x` এবং `y`, যা সংখ্যাত্মক মান ধারণ করে।

গ্রুপ (Group)

জাভাস্ক্রিপ্টের `Set` এর মতো একটি `Group` ক্লাস লিখুন। এর মেথডগুলি হবে: `add`, `delete`, এবং `has`।

ইটারেবল গ্রুপ (Iterable Group)

`Group` ক্লাসকে ইটারেবল (iterable) করে তুলুন।

অধ্যায় ৭: প্রজেক্ট (Project): একটি রোবট (A Robot)

> “মেশিন চিন্তা করতে পারে কি না [...] এই প্রশ্নটি সাবমেরিন সাঁতার কাটতে পারে কি না এই প্রশ্নের মতোই প্রাসঙ্গিক।”
>
> —এডসগার ডাইকস্ট্রা (Edsger Dijkstra), The Threats to Computing Science

মিডোফিল্ড (Meadowfield)

গ্রামটিতে কয়েকটি জায়গা আছে যা নিম্নলিখিত রাস্তাগুলি দ্বারা সংযুক্ত:

```
const roads = [
  "Alice's House-Bob's House",   "Alice's House-Cabin",
  "Alice's House-Post Office",   "Bob's House-Town Hall",
  "Daria's House-Ernie's House", "Daria's House-Town Hall",
  "Ernie's House-Grete's House", "Grete's House-Farm",
  "Grete's House-Shop",          "Marketplace-Farm",
  "Marketplace-Post Office",     "Marketplace-Shop",
  "Marketplace-Town Hall",       "Shop-Town Hall"
];
```

কাজটি (The Task)

রোবটের কাজ হল প্যাকেজগুলি (packages) সংগ্রহ এবং বিতরণ করা। এটি প্যাকেজগুলি কুড়িয়ে নেয় এবং সেগুলি তাদের গন্তব্যে (destination) পৌঁছে দেয়।

স্থায়ী ডাটা স্ট্রাকচার (Persistent Data Structure)

এই সিমুলেশনের জন্য আমরা স্থায়ী ডাটা স্ট্রাকচার (persistent data structures) ব্যবহার করব। এগুলি এমন অবজেক্ট যা কখনও পরিবর্তিত হয় না, তার বদলে নতুন অবস্থা (state) প্রয়োজন হলে নতুন অবজেক্ট তৈরি করে।

সিমুলেশন (Simulation)

রোবটের চলাচল সিমুলেট করার জন্য আমরা একটি `VillageState` ক্লাস তৈরি করব:

```
class VillageState {
  constructor(place, parcels) {
    this.place = place;
    this.parcels = parcels;
  }

  move(destination) {
    if (!roadGraph[this.place].includes(destination)) {
      return this;
    } else {
      let parcels = this.parcels.map(p => {
        if (p.place != this.place) return p;
```

```
        return {place: destination, address: p.address};
      }).filter(p => p.place != p.address);
      return new VillageState(destination, parcels);
    }
  }
}
```

পথ খোঁজা (Pathfinding)

রোবটকে সবচেয়ে কার্যকর পথ (efficient path) খুঁজে বের করতে হবে। এই পথ খোঁজার কৌশল ব্যবহার করে একটি লক্ষ্য-ভিত্তিক রোবট (goal-oriented robot):

```
function goalOrientedRobot({place, parcels}, route) {
  if (route.length == 0) {
    let parcel = parcels[0];
    if (parcel.place != place) {
      route = findRoute(roadGraph, place, parcel.place);
    } else {
      route = findRoute(roadGraph, place, parcel.address);
    }
  }
  return {direction: route[0], memory: route.slice(1)};
}
```

অনুশীলনী (Exercises)

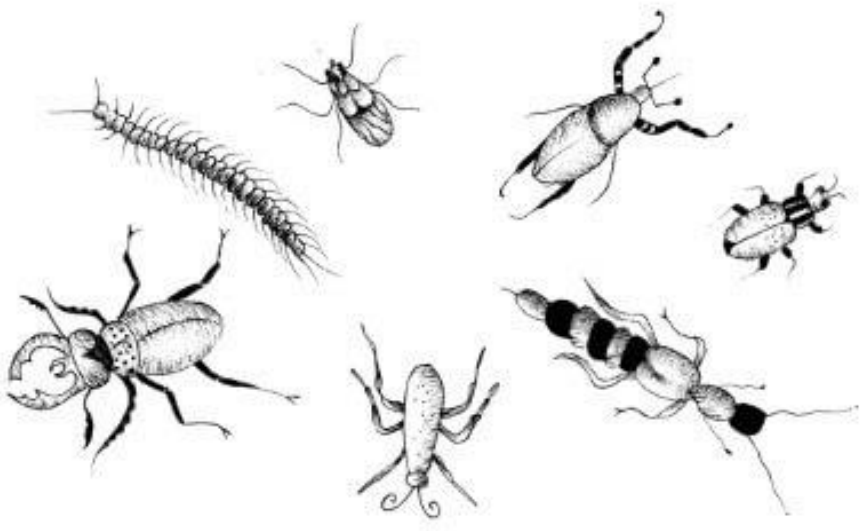
রোবট পরিমাপ (Measuring a Robot)

দুটি রোবটের কার্যকারিতা (efficiency) তুলনা করার জন্য একটি ফাংশন লিখুন।

রোবট দক্ষতা (Robot Efficiency)

আরও দক্ষ রোবট তৈরি করুন।

অধ্যায় ৮: বাগ (Bugs) এবং এরর (Errors)



বাগ এবং এররের চিত্র

> "ডিবাগিং কোড লেখার চেয়ে দ্বিগুণ কঠিন। তাই, আপনি যদি কোড যতটা সম্ভব চতুরতার সাথে লেখেন, তাহলে আপনি, সংজ্ঞা অনুযায়ী, এটি ডিবাগ করার জন্য যথেষ্ট স্মার্ট নন।"
>
> —ব্রায়ান কার্নিঘান (Brian Kernighan) এবং পি.জে. প্লগার (P.J. Plauger), The Elements of Programming Style

কঠোর মোড (Strict Mode)

JavaScript-কে আরও কঠোর করা যায় *স্ট্রিক্ট মোড (strict mode)* সক্রিয় করে। এটি করা যায় একটি ফাইল বা ফাংশন বডির শুরুতে `"use strict"` স্ট্রিং রেখে:

```
function canYouSpotTheProblem() {
  "use strict";
  for (counter = 0; counter < 10; counter++) {
    console.log("Happy happy");
  }
}

canYouSpotTheProblem();
// → ReferenceError: counter is not defined
```

টাইপ (Types)

কিছু ভাষা প্রোগ্রাম চালানোর আগেই আপনার সব বাইন্ডিং এবং এক্সপ্রেশনের টাইপ জানতে চায়। JavaScript শুধুমাত্র প্রোগ্রাম চলাকালীন টাইপ বিবেচনা করে। যদিও টাইপ প্রোগ্রাম সম্পর্কে কথা বলার জন্য একটি উপযোগী কাঠামো প্রদান করে।

টেস্টিং (Testing)

প্রোগ্রামে ত্রুটি খুঁজে বের করার সবচেয়ে সহজ উপায় হল এটি চালানো এবং এর আউটপুট পর্যবেক্ষণ করা। এটি হাতে বার বার করা একটি খারাপ ধারণা। এটি শুধু বিরক্তিকর নয়, এটি অকার্যকরও, কারণ প্রতিবার পরিবর্তন করার পর সবকিছু পরীক্ষা করতে অনেক সময় লাগে।

স্বয়ংক্রিয় টেস্টিং (automated testing):

```
function test(label, body) {
  if (!body()) console.log(`Failed: ${label}`);
}

test("convert Latin text to uppercase", () => {
  return "hello".toUpperCase() == "HELLO";
});
```

ডিবাগিং (Debugging)

একবার আপনি একটি বাগ (bug) নোটিশ করলে, এটি খুঁজে বের করতে হবে। কখনও কখনও এটি স্পষ্ট হয়, তবে অনেক সময় আপনি জানেন না কোথায় শুরু করবেন। ডিবাগিং (debugging) হল একটি প্রক্রিয়া যেখানে আপনি প্রোগ্রামের আচরণ বিশ্লেষণ করেন।

এক্সেপশন (Exceptions)

যখন একটি রানটাইম এরর (runtime error) ঘটে, JavaScript একটি এক্সেপশন (exception) তৈরি করে। এক্সেপশন হ্যান্ডলিং (exception handling) এর জন্য `try` এবং `catch` ব্যবহার করা হয়:

```
function promptNumber(question) {
  let result = Number(prompt(question));
  if (Number.isNaN(result)) throw new Error("Not a number");
  return result;
}

try {
  console.log(promptNumber("How many trees?"));
} catch (error) {
  console.log("That was not a valid number");
}
```

এক্সেপশনের পর পরিষ্কার করা (Cleaning Up After Exceptions)

কখনও কখনও আপনার প্রোগ্রামে এমন কোড থাকে যা এক্সেপশন ঘটলেও চালানো দরকার। `finally` ব্লক (block) ব্যবহার করে এটি করা যায়:

```
const account = {
  money: 100,
  transfer(amount) {
    if (amount > this.money) throw new Error("অপর্যাপ্ত অর্থ");
    this.money -= amount;
  }
};

function transfer(from, amount) {
  if (from.money < amount) throw new Error("অপর্যাপ্ত অর্থ");
  from.money -= amount;
  try {
    console.log("ট্রান্সফার সফল");
  } finally {
    from.money += amount;
  }
}
```

নির্বাচিত ক্যাচিং (Selective Catching)

প্রত্যাশিত সমস্যাগুলির জন্য, অধরা এক্সেপশন (uncaught exceptions) দিয়ে ক্র্যাশ করা একটি খারাপ কৌশল। কিন্তু যখন অপ্রত্যাশিত কিছু ভুল হয়, ক্র্যাশ করা সবচেয়ে ভাল অপশন।

```
class InputError extends Error {}

function promptDirection(question) {
  let result = prompt(question);
  if (result.toLowerCase() == "left") return "L";
  if (result.toLowerCase() == "right") return "R";
  throw new InputError("অবৈধ দিক: " + result);
}

for (;;) {
  try {
    let dir = promptDirection("কোন দিকে যেতে চান?");
    console.log("আপনি বেছে নিয়েছেন ", dir);
    break;
  } catch (e) {
    if (e instanceof InputError) {
      console.log("অনুগ্রহ করে সঠিক দিক লিখুন");
    } else {
      throw e;
    }
  }
}
```

অ্যাসারশন (Assertions)

অ্যাসারশন হল প্রোগ্রামের একটি চেক যা নির্দিষ্ট করে যে কিছু হওয়া উচিত। এটি ডিবাগিং এ সাহায্য করে:

```
function firstElement(array) {
  if (array.length == 0) {
    throw new Error("খালি অ্যারেতে firstElement কল করা হয়েছে");
  }
  return array[0];
}
```

অনুশীলনী (Exercises)

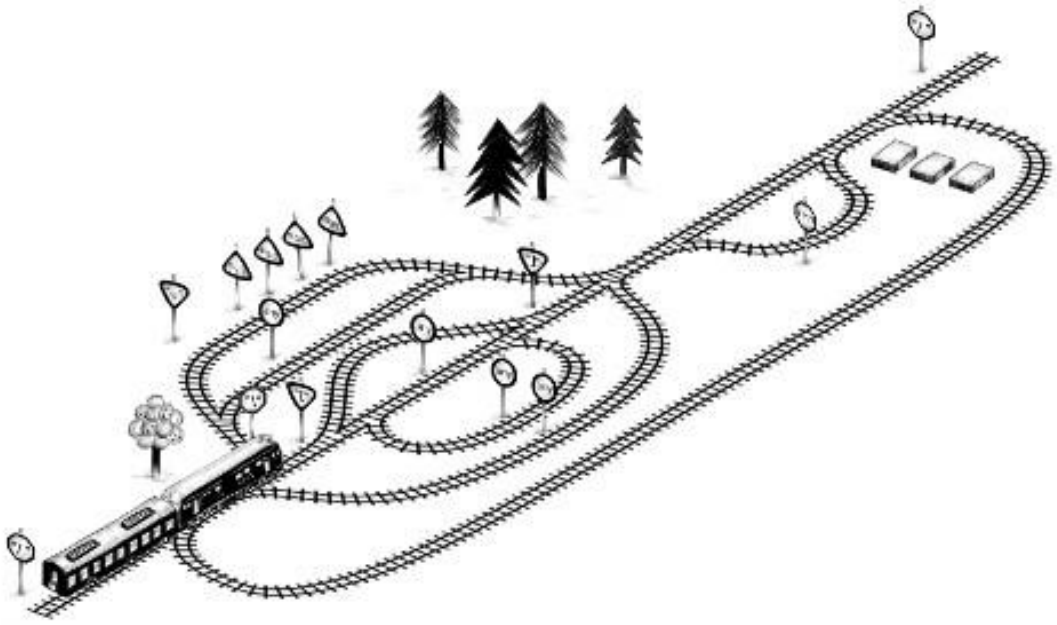
পুনরায় চেষ্টা (Retry)

কখনও কখনও একটি অপারেশন ব্যর্থ হতে পারে, কিন্তু পুনরায় চেষ্টা করলে সফল হতে পারে। একটি ফাংশন লিখুন যা একটি ফাংশনকে বারবার চেষ্টা করে যতক্ষণ না এটি সফল হয়।

লক করা বাক্স (The Locked Box)

একটি `withBoxUnlocked` ফাংশন লিখুন যা একটি ফাংশন ভ্যালু আর্গুমেন্ট (argument) হিসেবে নেয়, বাক্স আনলক করে, ফাংশন চালায়, এবং তারপর নিশ্চিত করে যে বাক্সটি আবার লক করা হয়েছে।

অধ্যায় ৯: রেগুলার এক্সপ্রেশন (Regular Expressions)



রেগুলার এক্সপ্রেশনের চিত্র

> "কিছু লোক যখন কোনো সমস্যার সম্মুখীন হয়, তারা ভাবে, 'আমি জানি, আমি রেগুলার এক্সপ্রেশন ব্যবহার করব।' এখন তাদের দুটি সমস্যা আছে।"
>
> —জেমি জাউইনস্কি (Jamie Zawinski)

টেক্সট প্যাটার্ন (Text Patterns)

রেগুলার এক্সপ্রেশন (regular expressions) হল টেক্সট প্যাটার্ন (text patterns) বর্ণনা করার একটি উপায়। এগুলি একটি বিশেষ মিনি-প্রোগ্রামিং ভাষা (mini programming language) ব্যবহার করে যা স্ট্রিং খোঁজার (searching) এবং প্রতিস্থাপন করার (replacing) জন্য ব্যবহৃত হয়।

JavaScript-এ রেগুলার এক্সপ্রেশন একটি অবজেক্ট (object):

```
let re1 = new RegExp("abc");
let re2 = /abc/;
```

প্যাটার্ন ম্যাচিং (Pattern Matching)

রেগুলার এক্সপ্রেশনের সবচেয়ে সাধারণ মেথড হল `test`। এটি একটি স্ট্রিং নেয় এবং প্যাটার্ন পাওয়া গেলে `true` রিটার্ন করে:

```
console.log(/abc/.test("abcde"));
// → true
```

```
console.log(/abc/.test("abxde"));
// → false
```

সেট অফ ক্যারেক্টারস (Sets of Characters)

বর্গ ব্র্যাকেট (square brackets) দিয়ে আপনি একাধিক ক্যারেক্টারের সেট নির্দিষ্ট করতে পারেন:

```
console.log(/[0123456789]/.test("in 1992"));
// → true
console.log(/[0-9]/.test("in 1992"));
// → true
```

পুনরাবৃত্তি (Repetition)

আপনি একটি প্যাটার্নের একাধিক ঘটনা ম্যাচ করতে পারেন:

```
// + অন্তত একবার
console.log(/'\d+'/.test("'123'"));
// → true

// * শূন্য বা তার বেশি বার
console.log(/'\d*'/.test("''"));
// → true

// ? ঐচ্ছিক (শূন্য বা একবার)
console.log(/neighbou?r/.test("neighbour"));
// → true
console.log(/neighbou?r/.test("neighbor"));
// → true
```

গ্রুপিং (Grouping)

প্যারেনথেসিস (parentheses) দিয়ে একটি প্যাটার্নের অংশগুলি গ্রুপ করা যায়:

```
let quotedText = /'([^']*)'/;
console.log(quotedText.exec("she said 'hello'"));
// → ["'hello'", "hello"]
```

অনুশীলনী (Exercises)

রেজেক্স গল্ফ (Regex Golf)

সংক্ষিপ্ততম রেগুলার এক্সপ্রেশন লিখুন যা নির্দিষ্ট প্যাটার্ন ম্যাচ করে।

কোটেশন স্টাইল (Quoting Style)

স্ট্রিংয়ের সব সিঙ্গেল কোট ডাবল কোটে পরিবর্তন করুন।

অধ্যায় ১০: মডিউল (Modules)

> “একটি প্রোগ্রামকে মডিউলে ভাগ করা সবচেয়ে গুরুত্বপূর্ণ। মডিউল একে অপরের থেকে স্বাধীন হওয়া উচিত। প্রত্যেকের নিজস্ব কাজ থাকা উচিত।”
>
> —টেফ (tef), programming is terrible

মডিউল স্কোপ (Module Scope)

মডিউলের বিশেষ বৈশিষ্ট্য হল এর নিজস্ব স্কোপ (scope) থাকে। একটি মডিউলে ডিক্লেয়ার করা ভেরিয়েবল অন্য মডিউলে দৃশ্যমান নয় যদি না এটি স্পষ্টভাবে এক্সপোর্ট (export) করা হয়।

প্যাকেজ (Packages)

একটি প্যাকেজ (package) হল একটি আলাদা মডিউল যা অন্যান্য প্রোগ্রামাররা ব্যবহার করতে পারে। JavaScript জগতে, NPM (*https://npmjs.org*) প্যাকেজ ম্যানেজার (package manager) হিসেবে ব্যবহৃত হয়।

ES মডিউলস (ES Modules)

JavaScript-এর আধুনিক সংস্করণে মডিউল সিস্টেম (module system) অন্তর্ভুক্ত করা হয়েছে। এটি `import` এবং `export` স্টেটমেন্ট ব্যবহার করে:

```
// mathUtils.js এ এক্সপোর্ট করা
export function square(x) {
  return x * x;
}

// অন্য ফাইলে ইমপোর্ট করা
import {square} from "./mathUtils.js";
console.log(square(11));
// → 121
```

CommonJS মডিউলস

Node.js এর প্রাথমিক মডিউল সিস্টেম CommonJS। এটি `require` এবং `module.exports` ব্যবহার করে:

```
// এক্সপোর্ট করা
module.exports = function formatDate(date, format) {
  return format.replace(/YYYY|M(MMM)?|Do?|dddd/g, tag => {
    if (tag == "YYYY") return date.getFullYear();
    if (tag == "M") return date.getMonth();
    if (tag == "MMMM") return months[date.getMonth()];
    if (tag == "D") return date.getDate();
    if (tag == "Do") return ordinal(date.getDate());
    if (tag == "dddd") return days[date.getDay()];
```

```
  });
};

// ইমপোর্ট করা
const formatDate = require("./format-date");
```

বিল্ডিং এবং বান্ডলিং (Building and Bundling)

একটি ওয়েব পেজে 200টি আলাদা ফাইল লোড করা সময়সাপেক্ষ। তাই ওয়েব প্রোগ্রামাররা বান্ডলার (bundler) ব্যবহার করে - এগুলি সব মডিউল একত্রিত করে একটি বড় ফাইলে পরিণত করে।

অনেক JavaScript প্যাকেজ TypeScript এর মতো ভাষা এক্সটেনশনে (language extensions) লেখা হয়। এগুলি প্লেইন JavaScript-এ কম্পাইল (compile) করা হয়।

মডিউল ডিজাইন (Module Design)

একটি ভাল মডিউল ডিজাইনের বৈশিষ্ট্য: - সহজে ব্যবহারযোগ্য ইন্টারফেস (interface) - বিদ্যমান কনভেনশন অনুসরণ - সরল ডাটা স্ট্রাকচার ব্যবহার - একটি নির্দিষ্ট কাজে ফোকাস - অন্য কোডের সাথে সহজে কম্পোজ করা যায়

অনুশীলনী (Exercises)

মডিউলার রোবট (Modular Robot)

রোবট সিমুলেশন প্রোজেক্টকে মডিউলে ভাগ করুন।

রোডস মডিউল (Roads Module)

রাস্তার গ্রাফ বিল্ড করার কোড একটি মডিউলে মুভ করুন।

অধ্যায় ১১: অ্যাসিনক্রোনাস প্রোগ্রামিং (Asynchronous Programming)

> “প্রতিশ্রুতি (Promises) বাস্তবে যেমন, প্রোগ্রামিং-এও তেমন - এগুলি ভাঙ্গা খুব সহজ।”
>
> —জন ফ্রেডরিক (John Frederick)

কলব্যাক (Callbacks)

JavaScript-এ অ্যাসিনক্রোনাস (asynchronous) প্রোগ্রামিং এর সবচেয়ে প্রাথমিক রূপ হল কলব্যাক (callbacks):

```
setTimeout(() => console.log("টাইমার শেষ!"), 2000);
console.log("টাইমার শুরু");

// → টাইমার শুরু
// → টাইমার শেষ!
```

প্রমিস (Promises)

প্রমিস (promises) হল অ্যাসিনক্রোনাস অপারেশনের ফলাফল প্রতিনিধিত্বকারী অবজেক্ট:

```
let promise = fetch("example.json");
promise.then(response => console.log("ডাটা এসেছে!"));

console.log("ডাটা লোড হচ্ছে...");
```

এসিনক এবং এওয়েট (Async and Await)

`async` এবং `await` কীওয়ার্ড অ্যাসিনক্রোনাস কোড লেখার একটি সহজ উপায় প্রদান করে:

```
async function fetchData() {
  try {
    let response = await fetch("example.json");
    let data = await response.json();
    console.log(data);
  } catch (error) {
    console.log("ত্রুটি:", error);
  }
}
```

জেনারেটর (Generators)

জেনারেটর (generators) হল এমন ফাংশন যা একাধিক মান উৎপন্ন করতে পারে:

```
function* numberGenerator() {
  yield 1;
  yield 2;
```

```
  yield 3;
}

let gen = numberGenerator();
console.log(gen.next().value);  // → 1
console.log(gen.next().value);  // → 2
console.log(gen.next().value);  // → 3
```

ইভেন্ট লুপ (Event Loop)

JavaScript একটি সিঙ্গেল থ্রেডেড (single-threaded) ভাষা, কিন্তু এর ইভেন্ট লুপ (event loop) অ্যাসিনক্রোনাস প্রোগ্রামিং সম্ভব করে:

```
console.log("প্রথম");

setTimeout(() => {
  console.log("তৃতীয়");
}, 0);

console.log("দ্বিতীয়");

// → প্রথম
// → দ্বিতীয়
// → তৃতীয়
```

অ্যাসিনক্রোনাস বাগ (Asynchronous Bugs)

অ্যাসিনক্রোনাস কোডে বাগ খুঁজে বের করা কঠিন হতে পারে: - টাইমিং নির্ভর বাগ - রেস কন্ডিশন (race conditions) - এরর হ্যান্ডলিং জটিল হতে পারে

অনুশীলনী (Exercises)

বিলম্বিত সময় (Delayed Time)

একটি `wait` ফাংশন লিখুন যা প্রমিস রিটার্ন করে এবং নির্দিষ্ট মিলিসেকেন্ড অপেক্ষা করে।

রিয়েল প্রমিস (Real Promise)

একটি প্রমিস ক্লাস বাস্তবায়ন করুন যা প্রমিস/A+ স্পেসিফিকেশন মেনে চলে।

কনকারেন্ট প্রমিস (Concurrent Promises)

একসাথে একাধিক প্রমিস চালানোর জন্য একটি ফাংশন লিখুন।

অধ্যায় ১২: জাভাস্ক্রিপ্ট এবং ব্রাউজার (JavaScript and the Browser)

> “পরস্পর সংযুক্ত ডকুমেন্টের একটি বিশ্বব্যাপী স্পেস কল্পনা করুন... যেখানে আপনি একটি লিঙ্ক ক্লিক করে অন্য ডকুমেন্টে যেতে পারবেন।”
>
> —টিম বার্নার্স-লি (Tim Berners-Lee)

নেটওয়ার্ক এবং ইন্টারনেট (Networks and the Internet)

ব্রাউজার আমাদেরকে দূরবর্তী কম্পিউটারে যাওয়ার অনুমতি দেয়। এই দূরবর্তী সংযোগ প্রতিষ্ঠার জন্য ব্রাউজার টিসিপি/আইপি (TCP/IP) প্রোটোকল (protocol) ব্যবহার করে।

ওয়েব (The Web)

একটি URL (Uniform Resource Locator) এর গঠন:

```
  http://eloquentjavascript.net/13_browser.html
 |      |                       |               |
protocol       server                path
```

HTML (HyperText Markup Language)

HTML হল ওয়েব পেজের জন্য ব্যবহৃত ডকুমেন্ট ফরম্যাট (document format)। একটি HTML ডকুমেন্টে টেক্সট এবং ট্যাগ থাকে যা টেক্সটকে কাঠামো দেয়:

```
<!doctype html>
<html>
  <head>
    <meta charset="utf-8">
    <title>মাই হোম পেজ</title>
  </head>
  <body>
    <h1>স্বাগতম!</h1>
    <p>এটি আমার ওয়েবসাইট</p>
  </body>
</html>
```

HTML এবং জাভাস্ক্রিপ্ট (HTML and JavaScript)

জাভাস্ক্রিপ্ট কোড HTML এ যোগ করার জন্য `<script>` ট্যাগ ব্যবহার করা হয়:

```
<h1>টেস্টিং অ্যালার্ট</h1>
<script>alert("হ্যালো!");</script>
```

বাইরের ফাইল থেকে স্ক্রিপ্ট লোড করা:

```
<h1>টেস্টিং অ্যালার্ট</h1>
<script src="code/hello.js"></script>
```

স্যান্ডবক্স (Sandbox)

ব্রাউজার নিশ্চিত করে যে ওয়েবসাইটে চলা জাভাস্ক্রিপ্ট কম্পিউটারে ক্ষতি করতে না পারে: - নিরাপত্তা নিশ্চিত করার জন্য স্যান্ডবক্সড পরিবেশে চলে - ফাইল সিস্টেম অ্যাক্সেস সীমিত - নেটওয়ার্ক অ্যাক্সেস সীমিত

কম্প্যাটিবিলিটি এবং ব্রাউজার ওয়ার্স (Compatibility and Browser Wars)

ব্রাউজার কম্প্যাটিবিলিটির (compatibility) প্রধান বৈশিষ্ট্য: - কম্প্যাটিবিলিটি ভালো - বাগ কম - ক্রস-ব্রাউজার সাপোর্ট

অধ্যায় ১৩: ডকুমেন্ট অবজেক্ট মডেল (Document Object Model)

> “সূর্য চারপাশে ঘোরে, গ্রহগুলি নাচে, এবং DOM ট্রি আমাদের সবাইকে একত্রে ধরে রাখে।”
>
> —অজানা ওয়েব ডেভেলপার

ডকুমেন্ট স্ট্রাকচার (Document Structure)

ব্রাউজার DOM (Document Object Model) কে একটি ট্রি স্ট্রাকচার (tree structure) হিসেবে প্রতিনিধিত্ব করে। এই ট্রি HTML ডকুমেন্টের স্ট্রাকচারকে প্রতিফলিত করে। উদাহরণস্বরূপ:

```
<html>
  <head>
    <title>মাই ওয়েবসাইট</title>
  </head>
  <body>
    <h1>আমার হোম পেজ</h1>
    <p>আমি মারিজন (Marijn) এবং এটি আমার হোম পেজ।</p>
    <p>এছাড়াও আমি একটি বই লিখেছি! এটি পড়ুন
      <a href="http://eloquentjavascript.net">এখানে</a>।</p>
  </body>
</html>
```

এলিমেন্ট খুঁজে বের করা (Finding Elements)

DOM থেকে এলিমেন্ট খুঁজে পাওয়ার কয়েকটি উপায়:

```
document.getElementsByTagName("a")[0];  // ট্যাগ নাম (tag name) দিয়ে
document.getElementById("main");        // ID দিয়ে
document.getElementsByClassName("note"); // ক্লাস নাম (class name) দিয়ে
```

ডকুমেন্ট পরিবর্তন করা (Changing the Document)

DOM ট্রি পরিবর্তন করার মেথড:

```
document.createElement("p");     // নতুন এলিমেন্ট (element) তৈরি
node.appendChild(childNode);     // চাইল্ড (child) যোগ করা
node.removeChild(childNode);     // চাইল্ড মুছে ফেলা
node.replaceChild(new, old);    // চাইল্ড বদল করা
```

অ্যাট্রিবিউট (Attributes)

এলিমেন্টের অ্যাট্রিবিউট (attributes) পরিবর্তন:

```
element.getAttribute("href");
element.setAttribute("href", "https://example.com");
```

লেআউট (Layout)

এলিমেন্টের সাইজ (size) এবং পজিশন (position): - `offsetWidth`, `offsetHeight` - মোট সাইজ (total size) - `clientWidth`, `clientHeight` - ভিতরের সাইজ (inner size) - `getBoundingClientRect()` - স্ক্রিনে পজিশন (position on screen)

স্টাইলিং (Styling)

CSS স্টাইল (styles) পরিবর্তন:

```
element.style.color = "blue";
element.style.fontFamily = "Arial";
```

ক্যাসকেডিং স্টাইল (Cascading Style)

CSS সিলেক্টর (selectors) ব্যবহার:

```
.note { color: blue; }          /* ক্লাস সিলেক্টর (class selector) */
#main { font-size: 20px; }      /* ID সিলেক্টর (ID selector) */
p > a { color: red; }           /* চাইল্ড সিলেক্টর (child selector) */
```

অ্যানিমেশন (Animation)

এলিমেন্ট অ্যানিমেট (animate) করা:

```
requestAnimationFrame(function animate(time) {
  element.style.left = pos + "px";
  requestAnimationFrame(animate);
});
```

অনুশীলনী (Exercises)

টেবিল তৈরি করা (Building a Table)

DOM মেথড ব্যবহার করে একটি টেবিল তৈরি করুন।

এলিমেন্ট বাই ট্যাগ নেম (Elements by Tag Name)

`getElementsByTagName` এর নিজস্ব ভার্সন বাস্তবায়ন করুন।

বিড়ালের টুপি (The Cat's Hat)

একটি বিড়াল এবং তার টুপির অ্যানিমেশন তৈরি করুন।

অধ্যায় ১৪: ইভেন্ট হ্যান্ডলিং (Event Handling)

> “আপনার মনের উপর আপনার নিয়ন্ত্রণ আছে—বাইরের ঘটনার উপর নেই। এটা উপলব্ধি করুন, এবং আপনি শক্তি খুঁজে পাবেন।”
>
> —মার্কাস অরেলিয়াস (Marcus Aurelius)

ইভেন্ট হ্যান্ডলার (Event Handlers)

একটি ইভেন্ট হ্যান্ডলার (event handler) হল একটি ফাংশন যা নির্দিষ্ট ইভেন্ট (event) ঘটলে কল করা হয়:

```
window.addEventListener("click", () => {
  console.log("আপনি ক্লিক করেছেন!");
});
```

ইভেন্টের ধরন (Events Types)

বিভিন্ন ধরনের ইভেন্ট (events): - মাউস ইভেন্ট (mouse events): `click`, `mousemove`, `mousedown` - কীবোর্ড ইভেন্ট (keyboard events): `keydown`, `keypress`, `keyup` - ফোকাস ইভেন্ট (focus events): `focus`, `blur` - লোড ইভেন্ট (load events): `load`, `unload`, `beforeunload`

ইভেন্ট অবজেক্ট (Event Objects)

ইভেন্ট হ্যান্ডলার একটি ইভেন্ট অবজেক্ট (event object) পায় যা ইভেন্ট সম্পর্কে তথ্য ধারণ করে:

```
button.addEventListener("click", event => {
  console.log("মাউস পজিশন:", event.clientX, event.clientY);
});
```

ইভেন্ট প্রপাগেশন (Event Propagation)

ইভেন্ট DOM ট্রি-তে দুই ধাপে প্রবাহিত হয়: 1. ক্যাপচারিং ফেজ (capturing phase): উপর থেকে নীচে 2. বাবলিং ফেজ (bubbling phase): নীচে থেকে উপরে

ডিফল্ট অ্যাকশন (Default Actions)

কিছু ইভেন্টের ডিফল্ট অ্যাকশন (default actions) থাকে। আমরা `preventDefault` মেথড দিয়ে এগুলি বন্ধ করতে পারি:

```
link.addEventListener("click", event => {
  event.preventDefault();  // লিঙ্কে যাওয়া বন্ধ করা
});
```

কী ইভেন্ট (Key Events)

কীবোর্ড ইভেন্ট হ্যান্ডল করা:

```
window.addEventListener("keydown", event => {
  if (event.key == "Enter") {
    console.log("Enter চাপা হয়েছে!");
  }
});
```

অনুশীলনী (Exercises)

বেলুন (Balloon)

পর্দায় একটি বেলুন (🎈) আঁকুন যা উপর/নীচে তীর কী (arrow keys) টিপলে বড়/ছোট হয়।

মাউস ট্র্যাকিং (Mouse Tracking)

মাউস নড়াচড়ার সময় একটি লাল ডট (dot) মাউসকে অনুসরণ করুক।

ট্যাব মেনু (Tabbed Menu)

ট্যাব ইন্টারফেস তৈরি করুন যেখানে ট্যাবে ক্লিক করলে সংশ্লিষ্ট কনটেন্ট দেখা যায়।

অধ্যায় ১৫: HTTP এবং ফর্ম (HTTP and Forms)

> “HTTP এর মাধ্যমে ইন্টারনেট একটি দস্তাবেজের বিশ্বব্যাপী ওয়েব থেকে একটি ডেটা শেয়ারিং প্ল্যাটফর্মে পরিণত হয়েছে।”
>
> —টিম বার্নার্স-লি (Tim Berners-Lee)

প্রোটোকল (Protocol)

ব্রাউজার প্রথমে eloquentjavascript.net এর আইপি অ্যাড্রেস খুঁজে বের করে এবং পোর্ট ৮০ তে TCP কানেকশন খোলার চেষ্টা করে। একটি HTTP রিকোয়েস্টের (request) উদাহরণ:

```
GET /18_http.html HTTP/1.1
Host: eloquentjavascript.net
User-Agent: Your browser's name
```

সার্ভার উত্তর দেয়:

```
HTTP/1.1 200 OK
Content-Length: 87320
Content-Type: text/html
Last-Modified: Fri, 13 Oct 2023 10:05:41 GMT

<!doctype html>
... বাকি ডকুমেন্ট
```

ফেচ (Fetch)

`fetch` API ব্যবহার করে HTTP রিকোয়েস্ট পাঠানো:

```
fetch("example/data.txt").then(response => {
  console.log(response.status);
  // → 200
  console.log(response.headers.get("Content-Type"));
  // → text/plain
});
```

ফর্ম (Forms)

ফর্ম ইন্টারফেস (form interface) ব্যবহারকারীর ইনপুট সংগ্রহ করে:

```
<form method="GET" action="example/submit.html">
  <p><input type="text" name="name"> (নাম)</p>
  <p><input type="password" name="password"> (পাসওয়ার্ড)</p>
  <p><button type="submit">পাঠান</button></p>
</form>
```

ফোকাস (Focus)

ফর্ম কন্ট্রোলে ফোকাস (focus) এবং ব্লার (blur) ইভেন্ট:

```
let help = document.querySelector("#help");
let fields = document.querySelectorAll("input");
for (let field of Array.from(fields)) {
  field.addEventListener("focus", event => {
    help.textContent = field.getAttribute("data-help");
  });
  field.addEventListener("blur", event => {
    help.textContent = "";
  });
}
```

ফাইল ফিল্ড (File Fields)

ফাইল রিডিং (reading):

```
let input = document.querySelector("input[type=file]");
input.addEventListener("change", () => {
  if (input.files.length > 0) {
    let file = input.files[0];
    console.log("আপনি", file.name, "সিলেক্ট করেছেন");
    if (file.type) console.log("ফাইলের ধরন:", file.type);
  }
});
```

অনুশীলনী (Exercises)

কন্টেন্ট নেগোসিয়েশন (Content Negotiation)

সার্ভারের সাথে কন্টেন্ট নেগোসিয়েশন করে বিভিন্ন ফরম্যাটে ডাটা আদান-প্রদান করুন।

জাভাস্ক্রিপ্ট ওয়ার্কবেঞ্চ (JavaScript Workbench)

একটি ওয়েব ইন্টারফেস তৈরি করুন যেখানে জাভাস্ক্রিপ্ট কোড লিখে রান করা যায়।

অধ্যায় ১৬: প্রজেক্ট (Project): একটি প্ল্যাটফর্ম গেম (A Platform Game)

> “সমস্ত বাস্তবতাই একটি গেম।”
>
> —ইয়েন ব্যাংকস (Iain Banks)

গেম (The Game)

আমরা একটি প্ল্যাটফর্ম গেম (platform game) তৈরি করব যেখানে: - লাল এলাকাগুলি লাভা (lava) - প্লেয়ার বাম-ডান তীর কী দিয়ে হাঁটতে পারে - উপরের তীর কী দিয়ে লাফাতে পারে - সব কয়েন সংগ্রহ করলে লেভেল শেষ হয়

টেকনোলজি (Technology)

গেমটি তৈরি করতে ব্যবহার করা হবে: - DOM এলিমেন্ট (elements) দিয়ে গ্রাফিক্স - কীবোর্ড ইভেন্ট হ্যান্ডলিং (keyboard event handling) - কলিশন ডিটেকশন (collision detection)

লেভেল (Levels)

লেভেল ডিজাইন করার জন্য টেক্সট ফরম্যাট (text format):

```
......................
..#................#..
..#..............=.#..
..#.........o.o....#..
..#.@......#####...#..
..#####............#..
......#++++++++++++#..
......##############..
......................
```

যেখানে: - `.` = খালি জায়গা (empty space) - `#` = দেয়াল (wall) - `+` = লাভা (lava) - `@` = প্লেয়ার (player) - `o` = কয়েন (coin) - `=` = হরিজন্টাল মুভিং লাভা (horizontal moving lava) - `|` = ভার্টিকাল মুভিং লাভা (vertical moving lava) - `v` = ড্রিপিং লাভা (dripping lava)

অ্যাক্টর (Actors)

মুভিং এলিমেন্টগুলি (প্লেয়ার, কয়েন, লাভা) অ্যাক্টর (actor) হিসেবে মডেল করা হয়।

প্লেয়ার ক্লাস:

```
class Player {
  constructor(pos, speed) {
    this.pos = pos;
    this.speed = speed;
  }
```

```
  get type() { return "player"; }
}
```

ড্রয়িং (Drawing)

DOM এলিমেন্ট দিয়ে গেম রেন্ডার (render) করা:

```
class DOMDisplay {
  constructor(parent, level) {
    this.dom = elt("div", {class: "game"}, drawGrid(level));
    this.actorLayer = null;
    parent.appendChild(this.dom);
  }
}
```

মোশন এবং কলিশন (Motion and Collision)

কলিশন ডিটেকশন (collision detection):

```
Level.prototype.touches = function(pos, size, type) {
  // চেক করে একটি আয়তক্ষেত্র গ্রিড এলিমেন্ট স্পর্শ করে কিনা
}
```

অনুশীলনী (Exercises)

গেম ওভার (Game Over)

প্লেয়ার একটি নির্দিষ্ট সংখ্যক লাইফ (lives) নিয়ে শুরু করে এবং লাভা স্পর্শ করলে লাইফ হারায়।

পজ (Pause)

স্পেস বার চাপলে গেম পজ (pause) করার সিস্টেম যোগ করুন।

মনস্টার (A Monster)

লাভা ছাড়াও একটি মনস্টার (monster) যোগ করুন যা প্লেয়ারকে ধাওয়া করে।

অধ্যায় ১৭: ক্যানভাসে ড্রয়িং (Drawing on Canvas)

> “ড্রয়িং একটি প্রতারণা।”
>
> —এম.সি. এশার (M.C. Escher)

ক্যানভাস এলিমেন্ট (The Canvas Element)

HTML5 <canvas> এলিমেন্ট একটি ড্রয়িং এরিয়া (drawing area) প্রদান করে। এটি জাভাস্ক্রিপ্ট দিয়ে পিক্সেল-লেভেল (pixel-level) ড্রয়িং করার জন্য ব্যবহৃত হয়:

```
<canvas width="120" height="60"></canvas>
```

লাইন এবং সারফেস (Lines and Surfaces)

ক্যানভাস কনটেক্সট (context) ব্যবহার করে লাইন আঁকা:

```
let cx = canvas.getContext("2d");
cx.strokeStyle = "blue";
cx.moveTo(10, 10);
cx.lineTo(100, 50);
cx.stroke();
```

পাথ (Paths)

পাথ (paths) হল লাইনের শৃঙ্খলা যা একটি আকৃতি তৈরি করে:

```
cx.beginPath();
cx.moveTo(50, 10);
cx.lineTo(10, 70);
cx.lineTo(90, 70);
cx.closePath();
cx.stroke();
```

কার্ভ (Curves)

বেজিয়ার কার্ভ (Bezier curves) এবং আর্ক (arcs) আঁকা:

```
cx.beginPath();
cx.moveTo(10, 90);
cx.quadraticCurveTo(60, 10, 90, 90);
cx.lineTo(60, 10);
cx.closePath();
cx.stroke();
```

টেক্সট (Text)

ক্যানভাসে টেক্সট লেখা:

```
cx.font = "28px Georgia";
cx.fillStyle = "fuchsia";
cx.fillText("হ্যালো!", 10, 50);
```

ইমেজ (Images)

ক্যানভাসে ইমেজ আঁকা:

```
let img = document.createElement("img");
img.src = "hat.png";
img.addEventListener("load", () => {
  cx.drawImage(img, 0, 0);
});
```

ট্রান্সফরমেশন (Transformations)

ট্রান্সফরমেশন মেথড (transformation methods): - `translate` - পজিশন পরিবর্তন - `scale` - সাইজ পরিবর্তন - `rotate` - ঘোরানো

অনুশীলনী (Exercises)

পাই চার্ট (Pie Chart)

একটি পাই চার্ট (pie chart) আঁকুন যা ডাটা ভিজ্যুয়ালাইজ করে।

বাউন্সিং বল (A Bouncing Ball)

একটি বল আঁকুন যা স্ক্রিনে বাউন্স করে।

অধ্যায় ১৮: প্রজেক্ট (Project): একটি পিক্সেল আর্ট এডিটর (A Pixel Art Editor)

> “আমি আমার সামনে অনেক রং দেখি। আমি আমার খালি ক্যানভাস দেখি। তারপর, আমি রং প্রয়োগ করার চেষ্টা করি যেমন শব্দ কবিতা গঠন করে, যেমন নোট সংগীত গঠন করে।”
>
> —জোয়ান মিরো (Joan Miró)

কম্পোনেন্ট (Components)

আমাদের অ্যাপ্লিকেশন কয়েকটি মূল কম্পোনেন্টে (components) ভাগ করা হবে: - ড্রয়িং ক্যানভাস (drawing canvas) - টুল সিলেকশন (tool selection) - কালার পিকার (color picker) - সেভ এবং লোড (save and load)

স্টেট (State)

অ্যাপ্লিকেশন স্টেট (application state): - পিক্সেল ডাটা (pixel data) - বর্তমান টুল (current tool) - বর্তমান কালার (current color) - ক্যানভাস সাইজ (canvas size)

DOM বিল্ডিং (DOM Building)

ইন্টারফেস নির্মাণের জন্য একটি সাহায্যকারী ফাংশন:

```
function elt(type, props, ...children) {
  let dom = document.createElement(type);
  if (props) Object.assign(dom, props);
  for (let child of children) {
    if (typeof child != "string") dom.appendChild(child);
    else dom.appendChild(document.createTextNode(child));
  }
  return dom;
}
```

ড্রয়িং টুল (Drawing Tools)

বিভিন্ন ড্রয়িং টুল (drawing tools): - ড্র টুল (draw tool) - লাইন টুল (line tool) - রেক্ট্যাঙ্গল টুল (rectangle tool) - ফ্লাড ফিল টুল (flood fill tool)

আনডু হিস্টরি (Undo History)

অ্যাকশন হিস্টরি (action history) সংরক্ষণ করে পূর্ববর্তী অবস্থায় ফিরে যাওয়া:

```
class UndoHistory {
  constructor(state) {
    this.done = [state];
```

```
    this.undone = [];
  }
  pushState(state) {
    this.done.push(state);
    this.undone = [];
  }
}
```

অনুশীলনী (Exercises)

কীবোর্ড বাইন্ডিং (Keyboard Bindings)

অ্যাপ্লিকেশনে কীবোর্ড শর্টকাট যোগ করুন। টুলের নামের প্রথম অক্ষর টুলটি নির্বাচন করে, এবং ctrl-Z বা command-Z আনডু সক্রিয় করে।

দক্ষ ড্রয়িং (Efficient Drawing)

বড় ক্যানভাসের জন্য ড্রয়িং পারফরম্যান্স (performance) উন্নত করুন।

বৃত্ত (Circles)

বৃত্ত আঁকার একটি টুল যোগ করুন।

অধ্যায় ১৯: প্রজেক্ট (Project): দক্ষতা-শেয়ারিং ওয়েবসাইট (A Skill-Sharing Website)

ডিজাইন (Design)

ওয়েবসাইটের মূল বৈশিষ্ট্য: - টক লিস্টিং (talk listings) - কমেন্টিং (commenting) - নতুন টক যোগ করা - লং পোলিং আপডেট (long-polling updates)

HTTP ইন্টারফেস (HTTP Interface)

রিকোয়েস্টের ধরন: - GET /talks - সব টক পাওয়া - GET /talks/[title] - নির্দিষ্ট টক পাওয়া - PUT /talks/[title] - নতুন টক তৈরি - POST /talks/[title]/comments - কমেন্ট যোগ করা - DELETE /talks/[title] - টক মুছে ফেলা

সার্ভার (Server)

Node.js সার্ভার: - ফাইল সার্ভিং (file serving) - রাউটিং (routing) - টক ম্যানেজমেন্ট (talk management)

ক্লায়েন্ট (Client)

ক্লায়েন্ট অ্যাপ্লিকেশনে আছে: - HTML টেমপ্লেট - CSS স্টাইল - JavaScript কোড

অ্যাপ্লিকেশন স্টেট: - টকের তালিকা - ব্যবহারকারীর নাম

অধ্যায় ২০: নোড.জেএস (Node.js)

Node.js একটি জাভাস্ক্রিপ্ট রানটাইম (runtime) যা সার্ভার-সাইড প্রোগ্রামিং (server-side programming) এর জন্য ব্যবহৃত হয়।

ফাইল সিস্টেম (File System)

ফাইল অপারেশন:

```
const {readFile, writeFile} = require("fs").promises;

readFile("file.txt", "utf8")
  .then(text => console.log("ফাইল কনটেন্ট:", text));

writeFile("greet.txt", "হ্যালো!")
  .then(() => console.log("ফাইল লেখা হয়েছে"));
```

কমান্ড লাইন ইন্টারফেস (Command Line Interface)

Node.js প্রোগ্রামে কমান্ড লাইন আর্গুমেন্ট (command line arguments):

```
let args = process.argv.slice(2);
console.log("আর্গুমেন্ট:", args);
```

HTTP সার্ভার (HTTP Server)

একটি সাধারণ HTTP সার্ভার:

```
const {createServer} = require("http");
let server = createServer((request, response) => {
  response.writeHead(200, {"Content-Type": "text/plain"});
  response.write("হ্যালো সার্ভার!");
  response.end();
});
server.listen(8000);
```

প্যাকেজ ম্যানেজমেন্ট (Package Management)

npm (Node Package Manager) ব্যবহার:

```
{
  "name": "my-project",
  "version": "1.0.0",
  "dependencies": {
    "express": "^4.17.1"
```

```
    }
}
```

অনুশীলনী (Exercises)

সার্চ টুল (Search Tool)

grep এর মত একটি কমান্ড লাইন টুল তৈরি করুন।

ডিরেক্টরি তৈরি (Directory Creation)

ফাইল সার্ভারে MKCOL মেথড যোগ করুন।

ওয়েবে পাবলিক স্পেস (A Public Space on the Web)

ফাইল শেয়ারিং সার্ভার তৈরি করুন।

অধ্যায় ২১: প্রজেক্ট - দক্ষতা-শেয়ারিং ওয়েবসাইট

chapter_picture_21.jpg

> “যদি আপনার জ্ঞান থাকে, তবে অন্যদের তাদের মোমবাতি জ্বালাতে দিন এতে।”
>
> —মার্গারেট ফুলার

একটি দক্ষতা-শেয়ারিং মিটিং হল একটি ইভেন্ট যেখানে একই আগ্রহ নিয়ে মানুষ জড়ো হয় এবং তাদের জানা বিষয়ে ছোট, অনানুষ্ঠানিক প্রেজেন্টেশন দেয়।

ডিজাইন

এই প্রজেক্টে দুটি অংশ আছে: - সার্ভার অংশ (Node.js এ লেখা) → একটি HTTP সার্ভার যা ডেটা সংরক্ষণ করে - ক্লায়েন্ট অংশ (ব্রাউজারের জন্য লেখা) → ইউজার ইন্টারফেস প্রদান করে

সার্ভার সিস্টেমের ডেটা সংরক্ষণ করে এবং ক্লায়েন্টকে সরবরাহ করে। এছাড়াও ক্লায়েন্ট-সাইড সিস্টেমের ফাইলগুলি সরবরাহ করে।

একটি টক হল একটি প্রেজেন্টেশন যা একজন নির্দিষ্ট প্রেজেন্টার দ্বারা দেওয়া হয়। প্রতিটি টকে একটি শিরোনাম, একটি সারসংক্ষেপ এবং একটি কমেন্ট অ্যারে থাকে।

লং পোলিং

সার্ভার থেকে ক্লায়েন্টকে তাৎক্ষণিক আপডেট পাঠানোর জন্য একটি কানেকশন প্রয়োজন। কিন্তু ওয়েব ব্রাউজার ঐতিহ্যগতভাবে কানেকশন গ্রহণ করে না।

লং পোলিং এর বৈশিষ্ট্য: - ক্লায়েন্ট নিয়মিত সার্ভারকে নতুন তথ্যের জন্য জিজ্ঞাসা করে - সার্ভারের কাছে নতুন কিছু না থাকলে উত্তর দিতে দেরি করে - ক্লায়েন্ট সর্বদা একটি পোলিং রিকোয়েস্ট খোলা রাখে

কোড উদাহরণ: লং পোলিং ইমপ্লিমেন্টেশন

```
function handleTalks(request, response) {
  let handler = handlers[request.method]; // GET, PUT, DELETE ইত্যাদি মেথড হ্যান্ডলার
  let resolved = Promise.resolve();

  // বর্তমান জ্ঞাত ইভেন্টগুলি ট্র্যাক করা
  if (request.headers.has("if-none-match")) {
    let tag = /"(.*)"/.exec(request.headers.get("if-none-match"));
    if (tag && tag[1] == version) {
      resolved = waitForChanges(); // পরিবর্তন না হওয়া পর্যন্ত অপেক্ষা
```

```
    }
  }

  resolved
    .then(() => handler(request))
    .catch(error => {
      return { status: 500, body: error.toString() };
    })
    .then(({ body, status = 200, type = "application/json" }) => {
      response.status(status).type(type).json(body);
    });
}
```

এখানে মূল বিষয়গুলি: 1. `waitForChanges()` ফাংশন পরিবর্তন না হওয়া পর্যন্ত অপেক্ষা করে 2. `if-none-match` হেডার ব্যবহার করে ভার্সন চেক করা হয় 3. এরর হ্যান্ডলিং 500 স্ট্যাটাস কোড দিয়ে করা হয়

HTTP ইন্টারফেস

আমরা JSON ব্যবহার করব রিকোয়েস্ট এবং রেসপন্স বডির ফরম্যাট হিসেবে। ইন্টারফেস `/talks` পাথের উপর কেন্দ্রীভূত।

সার্ভার নিম্নলিখিত এন্ডপয়েন্টগুলি প্রদান করে:

- `GET /talks` - সব টক পেতে
- `GET /talks/[title]` - নির্দিষ্ট টক পেতে
- `PUT /talks/[title]` - নতুন টক তৈরি বা আপডেট করতে
- `DELETE /talks/[title]` - টক মুছে ফেলতে
- `POST /talks/[title]/comments` - কমেন্ট যোগ করতে

উদাহরণ: টক ডেটা স্ট্রাকচার

```
{
  "title": "ইউনিসাইকেল টিউনিং",
  "presenter": "জামাল",
  "summary": "স্টাইলের জন্য সাইকেল মডিফাই করা",
  "comments": []
}
```

→ এই JSON স্ট্রাকচার একটি টক অবজেক্ট দেখায়

সার্ভার

সার্ভারের প্রধান কাজগুলি: - স্ট্যাটিক ফাইল সার্ভ করা - টক ম্যানেজ করা (তৈরি, পড়া, আপডেট, ডিলিট) - কমেন্ট যোগ করা - লং পোলিং সাপোর্ট

ক্লায়েন্ট

ক্লায়েন্ট অ্যাপ্লিকেশনে আছে: - HTML টেমপ্লেট - CSS স্টাইল
- JavaScript কোড

অ্যাপ্লিকেশন স্টেট: - টকের তালিকা - ব্যবহারকারীর নাম

ক্লায়েন্ট-সাইড কোড

ব্রাউজারে টক লিস্ট আপডেট করার জন্য:

```
async function fetchOK(url, options) {
  let response = await fetch(url, options);
  if (response.status < 400) return response;
  throw new Error(response.statusText);
}

// টক লিস্ট লোড করা
async function loadTalkList() {
  let tag = undefined;
  for (;;) {
    try {
      let headers = tag ? {"if-none-match": tag} : {};
      let response = await fetchOK("/talks", {headers});
      tag = response.headers.get("ETag");
      displayTalks(await response.json());
    } catch (e) {
      console.log("নেটওয়ার্ক এরর: " + e);
      await new Promise(resolve => setTimeout(resolve, 500));
    }
  }
}
```

কোড ব্যাখ্যা: - `fetchOK` ফাংশন HTTP রিকুয়েস্ট পাঠায় এবং এরর চেক করে - `loadTalkList` একটি বারবার চলমান লুপে টক লিস্ট আপডেট করে - নেটওয়ার্ক এরর হলে 500ms পর আবার চেষ্টা করে

অনুশীলনী সমাধান

১. ডিস্ক পারসিস্টেন্স

```
const fs = require('fs');

function saveToFile(data) {
  fs.writeFileSync("talks.json", JSON.stringify(data));
}

function loadFromFile() {
```

```
  try {
    return JSON.parse(fs.readFileSync("talks.json", "utf8"));
  } catch (e) {
    return {};
  }
}
```

→ এই সমাধানে `fs` মডিউল ব্যবহার করে ডেটা ফাইলে সেভ করা হয়

২. কমেন্ট ফিল্ড রিসেট

```
function handleSubmit(event) {
  event.preventDefault();
  const form = event.target;
  const comment = form.elements.comment.value;

  submitComment(comment).then(() => {
    form.reset();  // ফর্ম রিসেট করা
  });
}
```

→ `form.reset()` কল করে কমেন্ট ফিল্ড খালি করা হয় ````

সূচি (Index)

প্রথম অংশ: ভাষা (Language)

দ্বিতীয় অংশ: ব্রাউজার (Browser)

তৃতীয় অংশ: নোড (Node)

প্রজেক্ট (Projects)