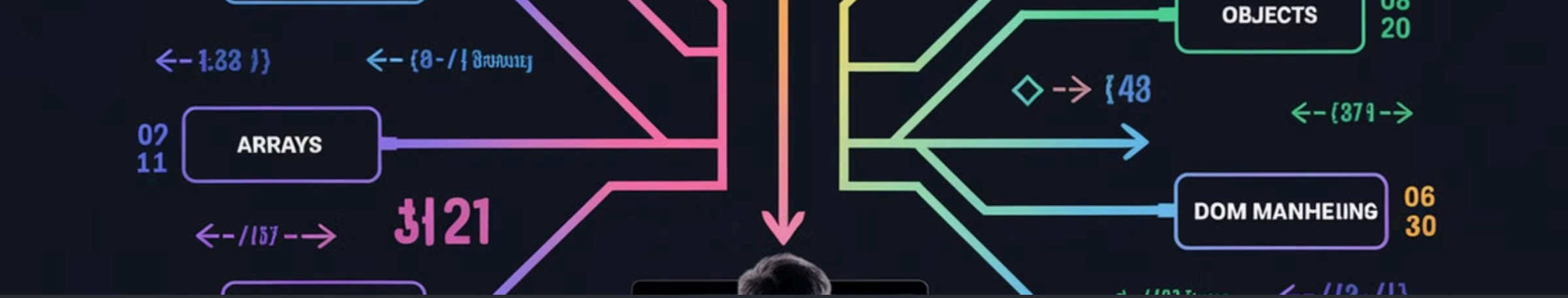
Основи JavaScript: Математика, Рядки та Логіка

Ласкаво просимо на другу лекцію курсу вивчення JavaScript! У цій презентації ми розглянемо базову математику, десяткові числа, залишок від ділення, скорочені оператори, роботу з рядками, логічні оператори та умовні конструкції.

Ці фундаментальні концепції є основою для розуміння мови програмування JavaScript та створення інтерактивних веб-додатків. Давайте почнемо наше занурення у світ JavaScript!

by o d



Зміст Лекції

1 **Базова математика**

Основні арифметичні операції в JavaScript: додавання, віднімання, множення та ділення. Порядок виконання операцій та використання операторів ++ і --.

2 **Десяткові числа та залишок від ділення**

Робота з числами з плаваючою комою, проблеми точності. Використання оператора % для отримання залишку від ділення.

3 **Скорочені оператори та робота з рядками**

Використання скорочених операторів +=, -=, *=, /=. Створення, об'єднання та маніпуляції з рядками.

4 **Логічні оператори та умовні конструкції**

Логічні значення, умовні оператори, оператори рівності, оператори І та АБО, тернарний оператор та оператор switch.

Базова математика в JavaScript

Основні операції

JavaScript підтримує стандартні математичні операції:

- додавання (+),
- віднімання (-),
- множення (*)
- ділення (/).

Порядок виконання операцій відповідає стандартним математичним правилам.

Приклади

- let a = 10;
- let b = 5;
- let result = a + b; // 15
- let result = a - b; // 5
- let result = a * b; // 50
- let result = a / b; // 2

Інкремент і декремент

Оператори ++ і -- дозволяють збільшувати або зменшувати значення змінної на 1:

let x = 5;

x++; // 6

x--; // 5

Десяткові числа в JavaScript

Підтримка чисел з плаваючою комою

JavaScript підтримує роботу з числами з плаваючою комою, що дозволяє виконувати обчислення з десятковими значеннями. Однак, при роботі з такими числами можуть виникати проблеми точності через особливості представлення чисел у комп'ютерній пам'яті.

Проблеми точності

```
let x = 0.1;

 let y = 0.2;

let res = x + y; // 0.30000000000000004
```

Це відома проблема в JavaScript і багатьох інших мовах програмування, пов'язана з двійковим представленням десяткових чисел.

Множення та ділення

```
let res = 0.1 * 0.2; // 0.020000000000000004

let res = 0.3 / 0.1; // 3
```

При роботі з фінансовими розрахунками рекомендується використовувати спеціальні бібліотеки для забезпечення точності.

Залишок від ділення

Оператор % повертає залишок від ділення одного числа на інше, що важливо для різноманітних алгоритмічних задач.

Оператор %

Обчислює залишок від ділення. Застосовується для перевірки парності/непарності чисел, циклічних операцій та інших алгоритмів.

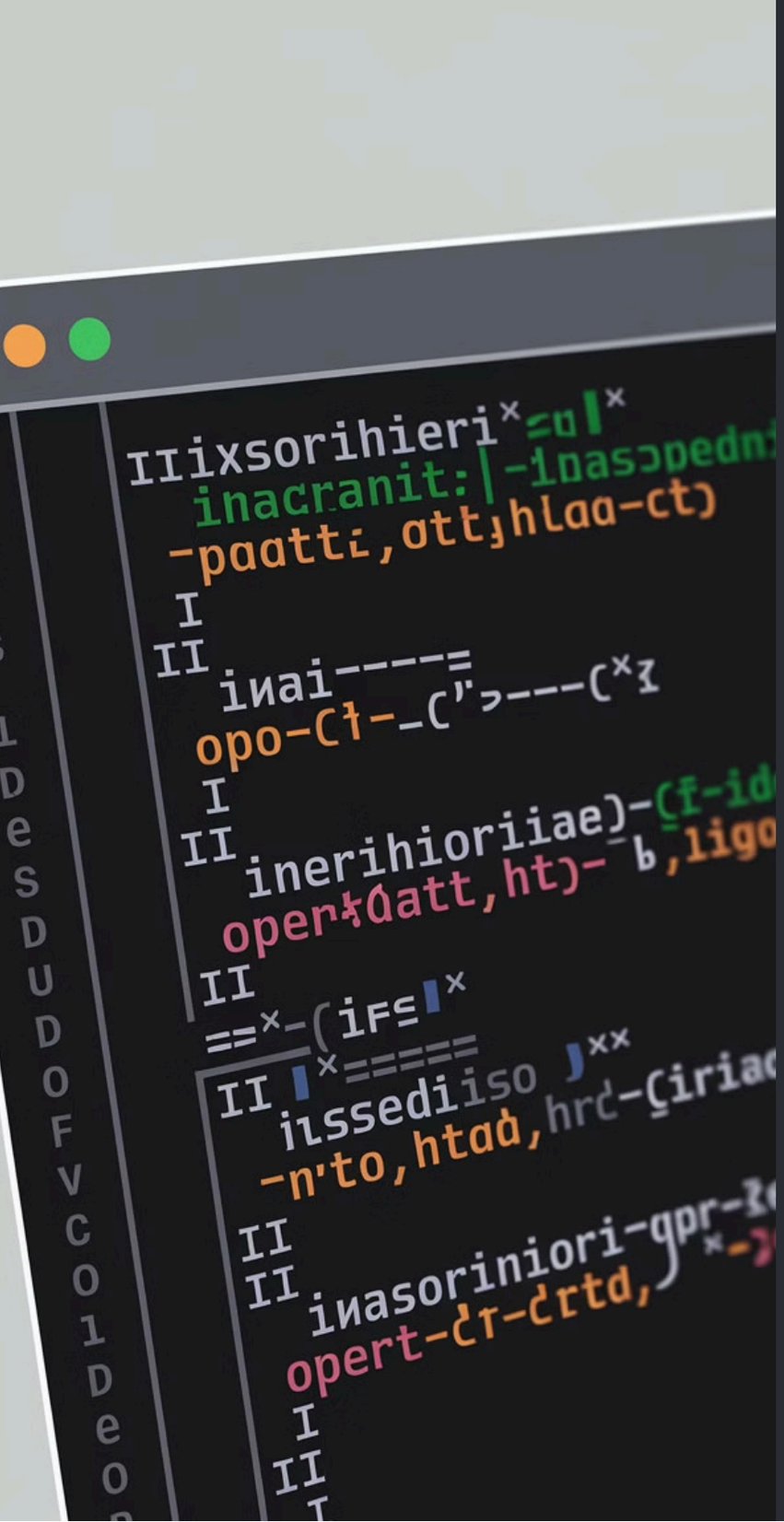
Приклади

10 % 3 = 1 (оскільки 10 = 3 × 3 + 1)

7 % 2 = 1 (непарне число, бо залишок від ділення на 2 дорівнює 1)

Практичне застосування

Створення циклічних послідовностей: визначення години доби (0-23) або дня тижня (0-6) після додавання певної кількості одиниць часу.



Скорочені оператори

1

Оператор +=

Додає значення до змінної та присвоює результат цій же змінній. Приклад: let y = 10; y += 5; // y тепер дорівнює 15 (еквівалентно y = y + 5)

2

Оператор -=

Віднімає значення від змінної та присвоює результат цій же змінній. Приклад: y -= 3; // y тепер дорівнює 12 (еквівалентно y = y - 3)

3

Оператор *=

Множить змінну на значення та присвоює результат цій же змінній. Приклад: y *= 2; // y тепер дорівнює 24 (еквівалентно y = y * 2)

4

Оператор /=

Ділить змінну на значення та присвоює результат цій же змінній. Приклад: y /= 4; // y тепер дорівнює 6 (еквівалентно y = y / 4)

Робота з рядками: Основи

1

Створення рядків

Рядки в JavaScript створюються за допомогою одинарних ('), подвійних ("") або зворотних (``) лапок. Приклад: let greeting = "Привіт"; let ім'я = "Світ";

2

Об'єднання рядків

Оператор + використовується для об'єднання (конкатенації) рядків. Приклад: let fullGreeting = str + ", " + ім'я + "!"; // "Привіт, Світ!"

3

Доступ до символів

Квадратні дужки використовуються для доступу до символів рядка за індексом (починаючи з 0). Приклад: let x = fullGreeting[0]; // "П"

4

Екранування символів

Для використання спеціальних символів, таких як лапки, використовується символ екранування (\). Приклад: let str = "Це \"цитата\" у рядку.";

Робота з рядками: Методи та властивості

Довжина рядка

Властивість **length** повертає кількість символів у рядку, включаючи пробіли та спеціальні символи.

Приклад:
let str = "Привіт, світ!";
console.log(str.length); // Виведе: 13

Методи зміни рядка

1. ### replace()

   let str = "Привіт, світ!".replace("світ", "JavaScript");

2. ### toUpperCase()

   let str = "привіт".toUpperCase(); // "ПРИВІТ"

3. ### toLowerCase()

   let str = "ПРИВІТ".toLowerCase(); // "привіт"

Методи пошуку

1. ### indexOf()

   "JavaScript".indexOf("Script"); // 4

2. ### includes()

   "Привіт".includes("ив"); // true

3. ### startsWith() і endsWith()

   "JavaScript".startsWith("Java"); // true
   "JavaScript".endsWith("Script"); // true

Шаблонні рядки

Шаблонні рядки (з використанням зворотних лапок `) дозволяють:

1. ### Вставляти змінні

   let name = "Олександр";
   let greeting = `Привіт, ${name}!`;

2. ### Багаторядкові рядки

   Створювати багаторядкові рядки без символів \n

3. ### Виконувати вирази

   `2 + 2 = ${2+2}`

Логічні значення та умовні оператори

Логічні оператори та умовні конструкції керують ходом виконання програми в JavaScript.

Логічні значення

Boolean значення (true/false) використовуються для порівнянь (5 > 3 = true) та керування програмою. Це один з основних типів даних JavaScript.

Оператор if

Виконує код лише за умови true. Приклад:

if (temperature > 30) { console.log("Спекотно!"); }. Умовою може бути порівняння або складніший вираз.

Оператор else

Виконує код, коли умова if є false. Приклад:

if (score >= 60) { console.log("Складено"); }

else { console.log("Перездача"); }

Оператор else if

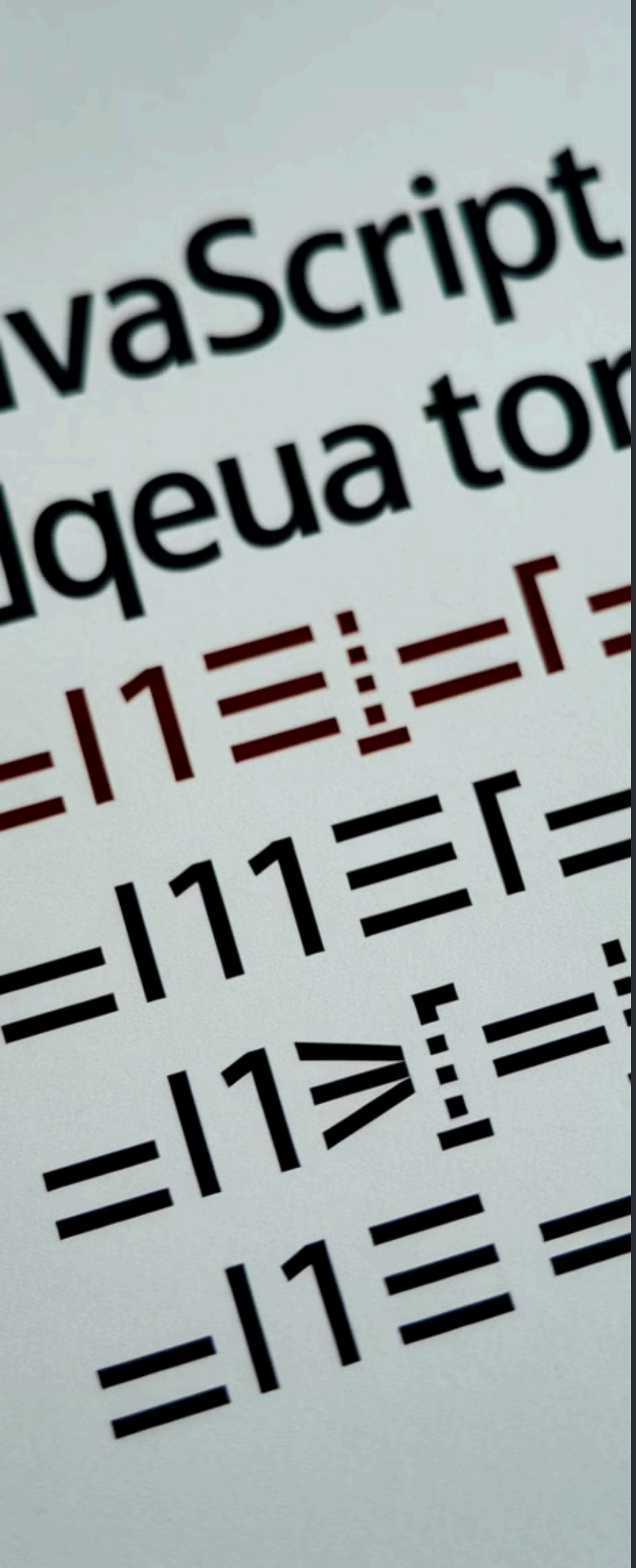
Перевіряє додаткові умови перед else. Приклад:

if (score >= 90) { console.log("Відмінно"); }

else if (score >= 75) { console.log("Добре"); }

else if (score >= 60) { console.log("Задовільно"); }

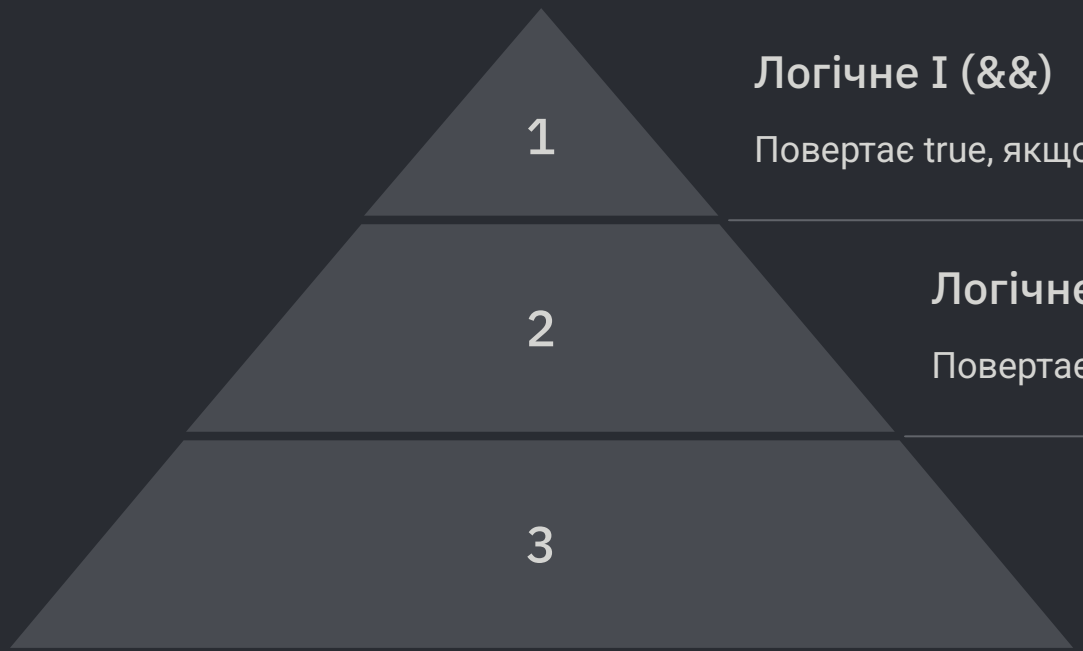
else { console.log("Незадовільно"); }



Оператори рівності

Оператор	Опис	Приклад	Результат
==	Рівність (перевіряє тільки значення)	5 == "5"	true
===	Сувора рівність (перевіряє значення та тип)	5 === "5"	false
!=	Нерівність (перевіряє тільки значення)	5 != "6"	true
!==	Сувора нерівність (перевіряє значення та тип)	5 !== "5"	true
>	Більше ніж	5 > 3	true
<	Менше ніж	5 < 3	false
>=	Більше або дорівнює	5 >= 5	true
<=	Менше або дорівнює	5 <= 3	false

Оператори І та АБО



Логічне І (&&)

Повертає true, якщо обидві умови істинні

Логічне АБО (||)

Повертає true, якщо хоча б одна умова істинна

Логічне НЕ (!)

Інвертує логічне значення

Оператори && (І) та || (АБО) дозволяють об'єднувати кілька умов в одному виразі. Оператор && повертає true, тільки якщо всі умови істинні, а оператор || повертає true, якщо хоча б одна умова істинна.

Приклади: let a = true; let b = false; console.log(a && b); // false console.log(a || b); // true

Оператор ! (НЕ) інвертує логічне значення: !true повертає false, а !false повертає true. Це корисно для перевірки протилежних умов.

Тернарний оператор

1

Синтаксис

умова ? вираз1 : вираз2

2

Принцип роботи

Якщо умова істинна, повертається вираз1, інакше - вираз2

3

Приклад використання

```
let age = 18;

let status = (age >= 18) ? "повнолітній" : "неповнолітній";
```

Тернарний оператор (? :) дозволяє скоротити умовну конструкцію if-else до одного рядка. Він особливо корисний, коли потрібно присвоїти змінній одне з двох значень залежно від умови.

Тернарний оператор можна використовувати для створення коротких і читабельних умовних виразів, але для складних умов краще використовувати традиційні if-else конструкції для збереження читабельності коду.

Оператор switch

Оператор switch — це конструкція, яка дозволяє перевіряти одну змінну на відповідність кільком можливим значенням. Замість багатьох if-else використовується більш компактна структура.

1

Оголошення змінної

let dayNumber = 3; let dayName;

2

Конструкція switch

switch (dayNumber) {

3

Перевірка випадків

case 1: dayName = "Понеділок"; break;
case 2: dayName = "Вівторок"; break;
case 3: dayName = "Середа"; break;

4

За замовчуванням

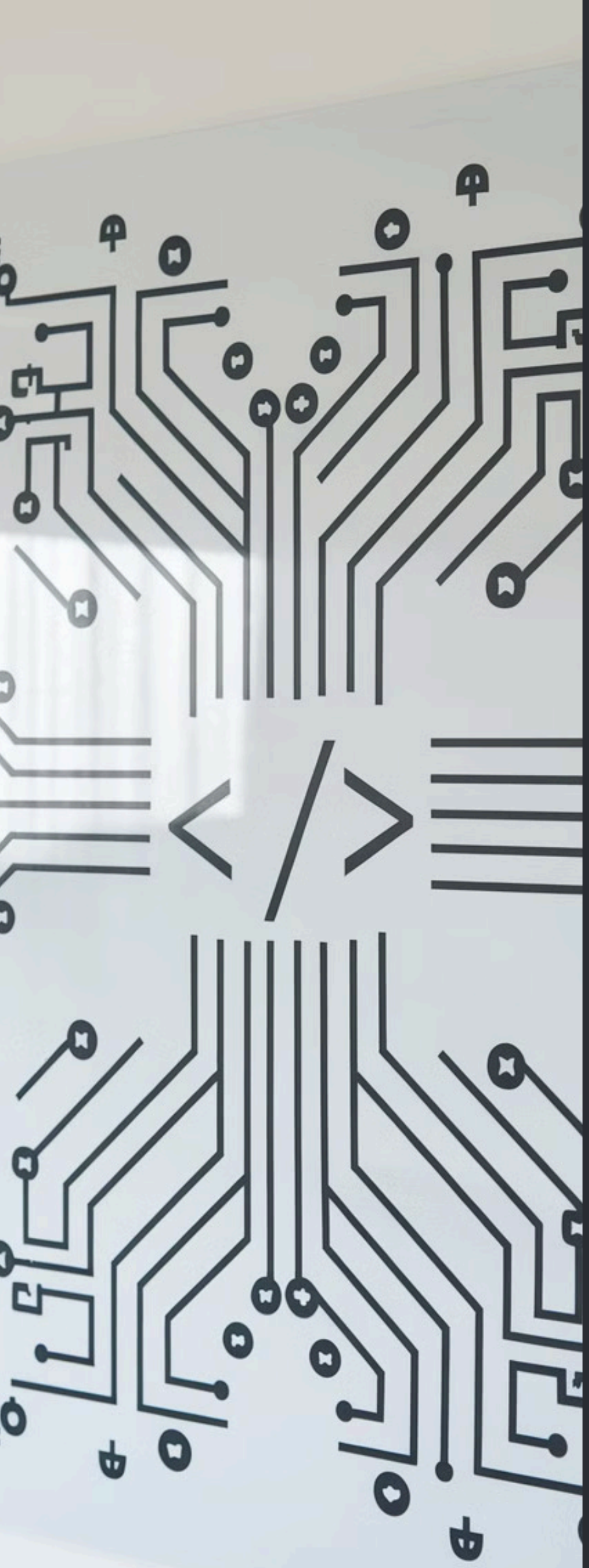
default: dayName = "Невідомий день";
}

5

Результат

console.log(dayName); // "Середа"

Ключове слово break необхідне для виходу з конструкції switch після виконання відповідного блоку коду. Без break виконання продовжиться до наступного case або default.



Підсумок лекції

1

Математичні операції

Ми вивчили основні арифметичні операції, роботу з десятковими числами та залишок від ділення.

2

Скорочені оператори

Розглянули скорочені оператори для оновлення значень змінних: +=, -=, *=, /=.

3

Рядки

Дослідили створення, об'єднання та маніпуляції з рядками, включаючи методи та властивості.

4

Логіка

Вивчили логічні оператори, умовні конструкції, оператори рівності та оператор switch.

Ці фундаментальні концепції є основою для розуміння JavaScript та створення інтерактивних веб-додатків. У наступних лекціях ми будемо будувати на цих знаннях, вивчаючи більш складні теми та практичні застосування.