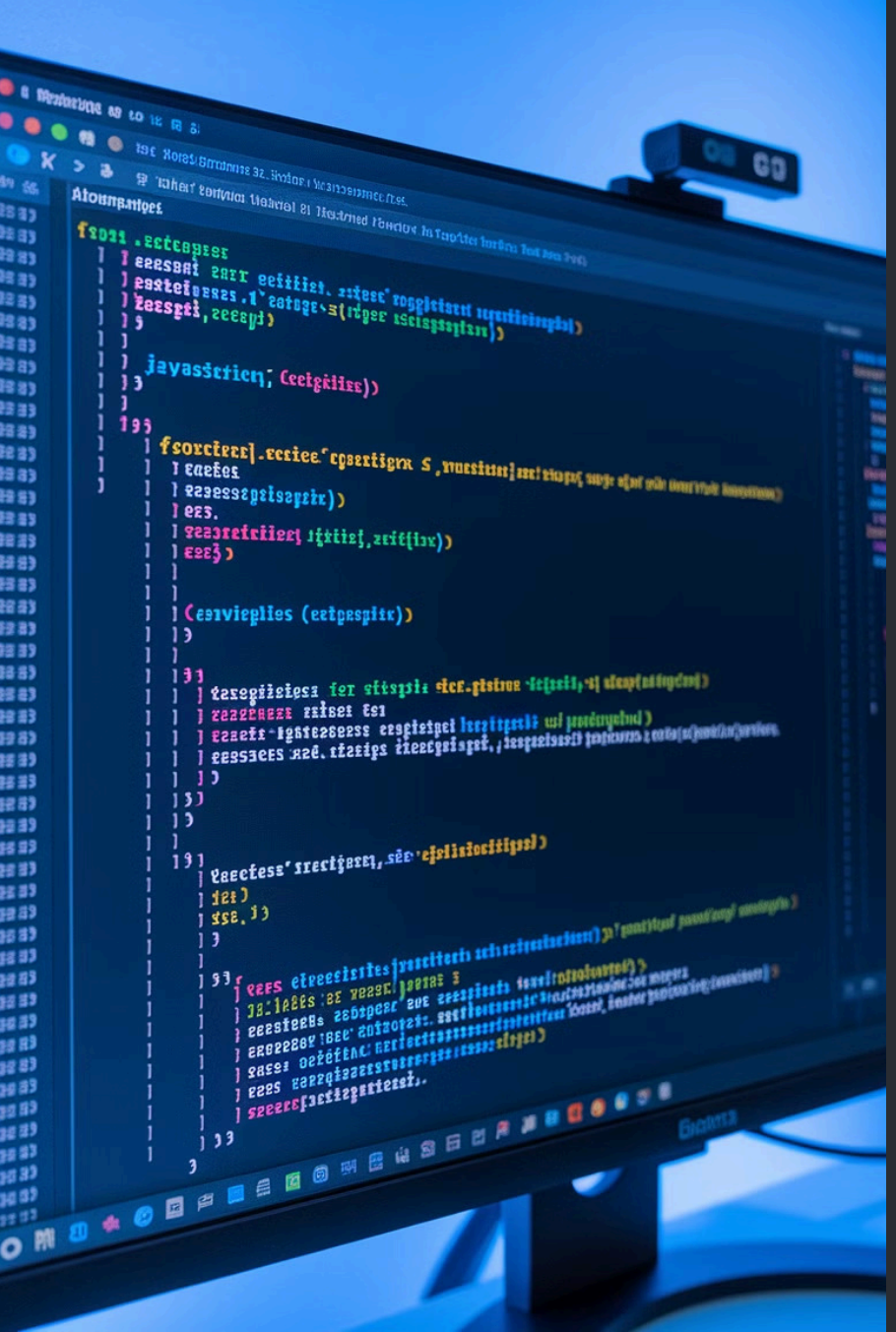


Основи JavaScript: Функції та Область Видимості

Ласкаво просимо на третю лекцію курсу вивчення JavaScript! Сьогодні ми заглибимось у важливі концепції, які є фундаментальними для розуміння мови програмування JavaScript: функції, аргументи, параметри за замовчуванням, область видимості та повернення значень.

Ці концепції є ключовими для створення ефективного, повторно використовуваного коду та розуміння того, як JavaScript обробляє дані у різних частинах вашої програми.

by o d

Що таке функції в JavaScript?

Функції - це повторно використовувані блоки коду, які інкапсулюють логіку програми.

1

Визначення

Функції дозволяють виконувати одну й ту саму дію в різних частинах програми без дублювання коду, що спрощує підтримку та тестування.

2

Синтаксис

Стандартний синтаксис:

function ім'яФункції(параметр1, параметр2)

{ /* тіло функції */ }.

Наприклад:

function calculateSum(a, b) { return a + b; }.

У ES6 також з'явилися стрілкові функції: **const add = (a, b) => a + b;**

3

Виклик

Функції викликаються за допомогою імені й дужок з аргументами: **calculateSum(5, 3);** поверне 8. Функції можуть викликатися з будь-якого місця коду, доступного їх області видимості, включаючи виклик з інших функцій.

Приклад простої функції

Створення функції

Спочатку ми визначаємо функцію з ім'ям "greeting", яка не приймає жодних параметрів:

```
function greeting() {
  console.log("Привіт, світ!");
}
```

Виклик функції

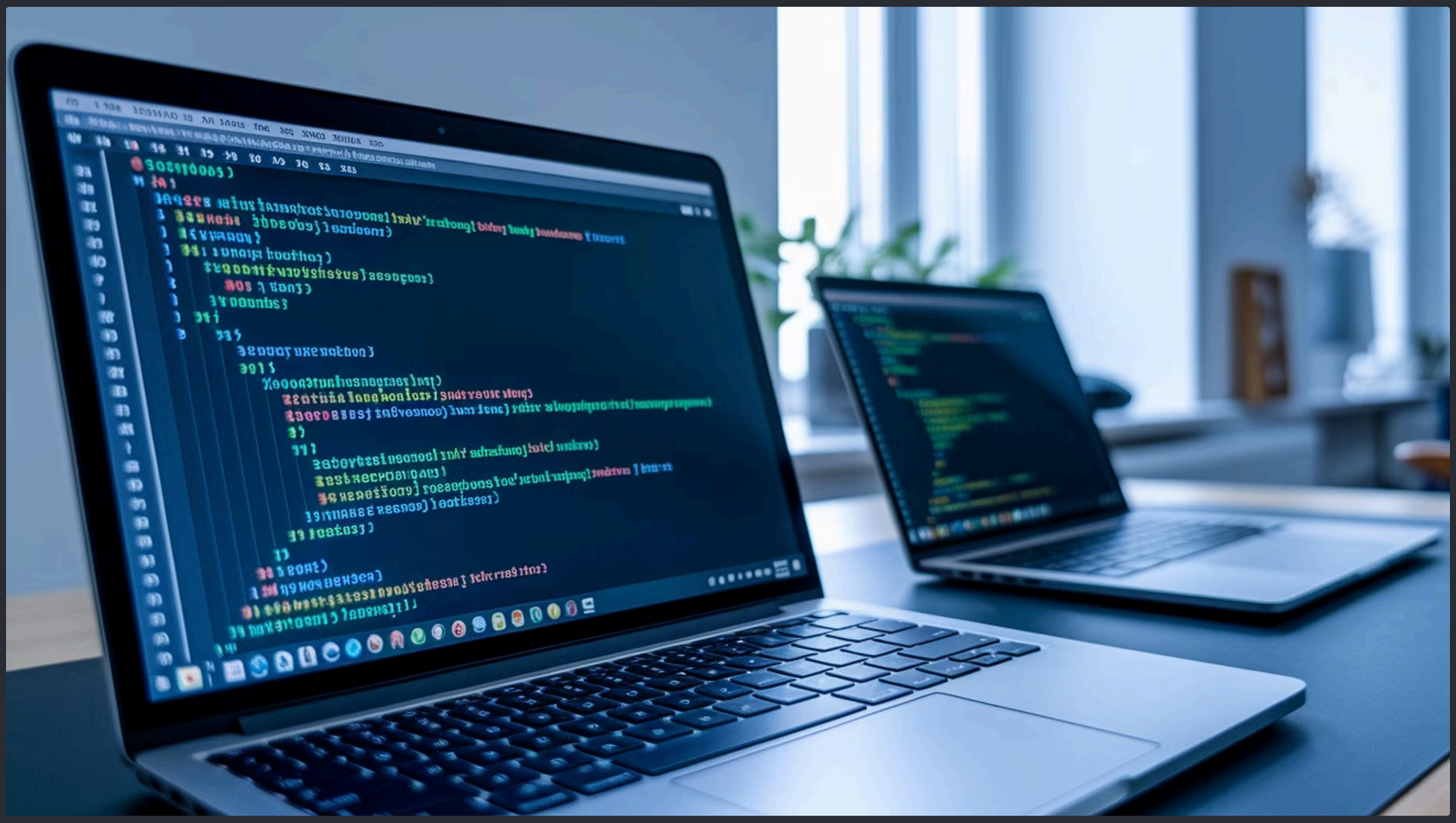
Після визначення функції ми можемо викликати її в будь-якому місці нашого коду:

```
greeting();
```

Результат

Коли функція викликається, вона виконує код у своєму тілі і виводить повідомлення:

// Виведе: Привіт, світ!



Аргументи функцій

1 Що таке аргументи?

Аргументи - це дані, що передаються функції при виклику для налаштування її поведінки.

2 Приклад з аргументом

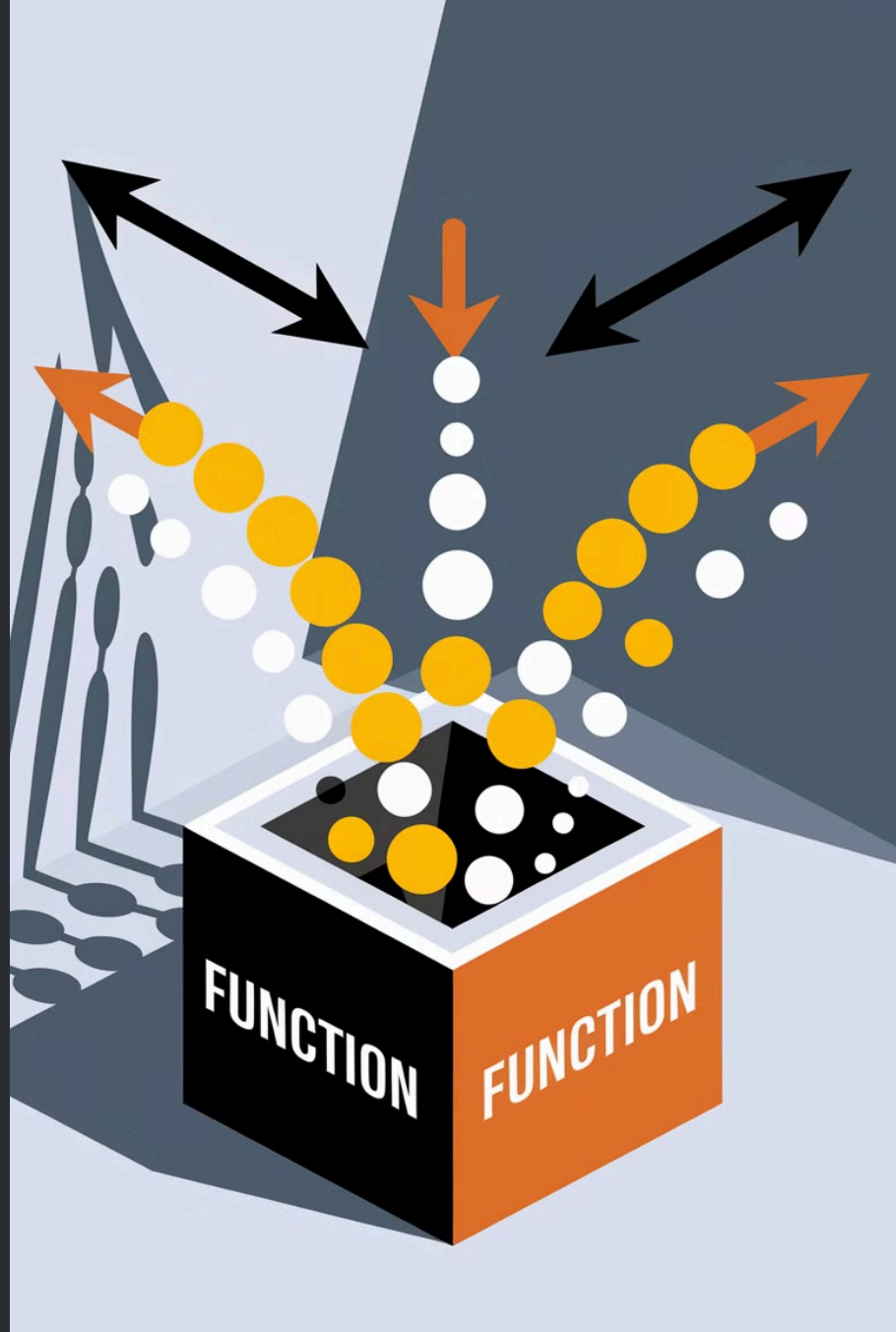
function greetingWithName(name) { console.log("Привіт, " + name + "!"); }

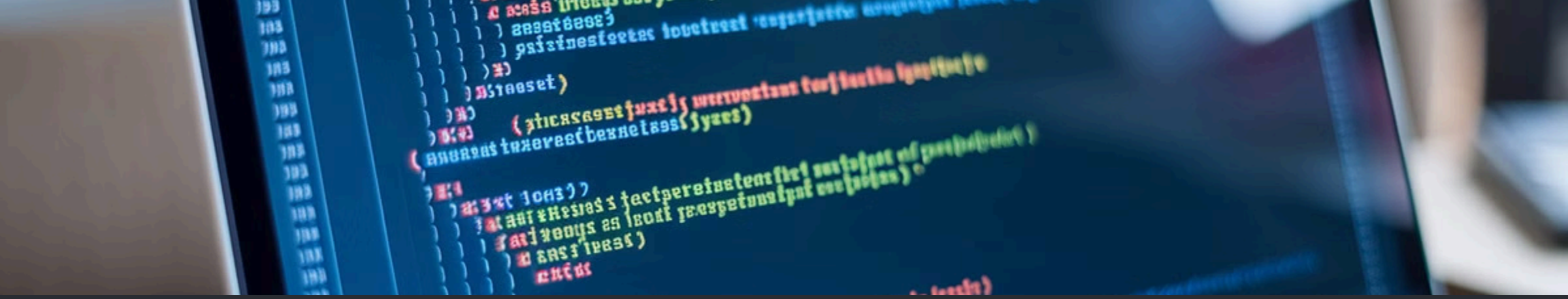
Тут **name** - параметр, який приймає значення при виклику.

3 Виклик з аргументом

greetingWithName("Аліса"); // Виведе: Привіт, Аліса!

"Аліса" - аргумент, що передається функції та присвоюється параметру **name**.





Параметри за замовчуванням

1 2 3

Визначення

Параметри за замовчуванням дозволяють задавати значення для параметрів функції, якщо вони не були передані під час виклику.

Синтаксис

function greetingWithDefaultParameter(name = "Світ") { ... }

Тут "Світ" - це значення за замовчуванням для параметра **name**.

Використання

Якщо аргумент не передано: **greetingWithDefaultParameter();** // Виведе: Привіт, Світ!

Якщо аргумент передано: **greetingWithDefaultParameter("Боб");** // Виведе: Привіт, Боб!

Глобальна область видимості

Визначення

Змінні, оголошені поза функціями (var), мають глобальну область видимості. Вони доступні для використання у будь-якій частині програми.

Приклад

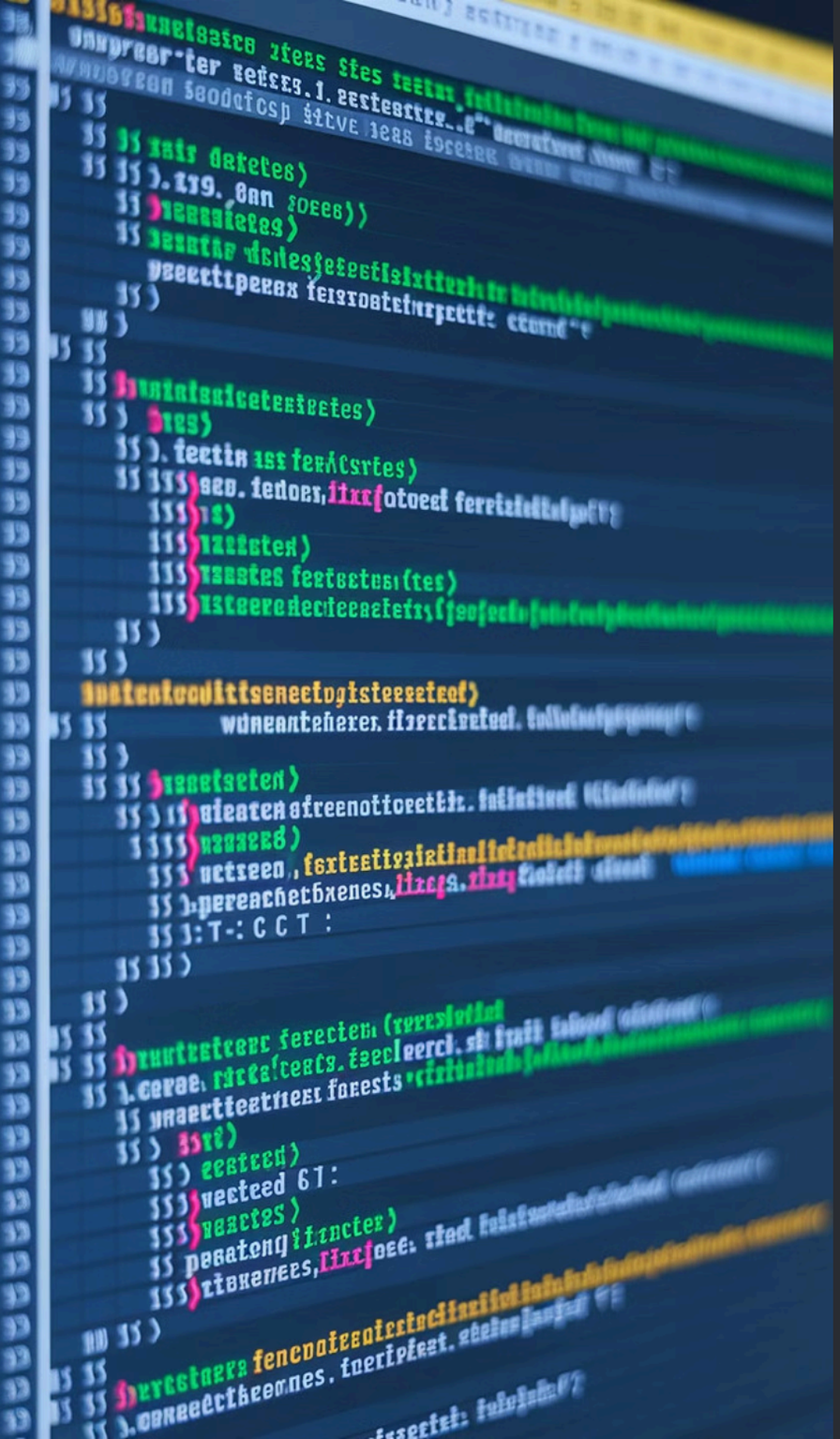
```
var globalVariable = "Я глобальна";

function showGlobalVariable() { console.log(globalVariable); }
```

Результат

```
showGlobalVariable();
// Виведе: Я глобальна
```

Змінна **globalVariable** доступна всередині функції, хоча була оголошена поза нею.



Локальна область видимості

1

Визначення

Змінні, оголошені всередині функції (let), мають локальну область видимості. Вони доступні лише в межах цієї функції та зникають після її завершення.

2

Приклад коду

function localFunction()

{ let localVariable = "Я локальна"; console.log(localVariable); }

3

Виклик функції

localFunction(); // Виведе: Я локальна

4

Спроба доступу ззовні

// console.log(localVariable); // Викличе помилку, оскільки localVariable не доступна поза функцією

Глобальна vs Локальна область видимості

Конфлікт імен

Коли у функції є змінні з однаковою назвою, що існують у глобальній і локальній області, пріоритет має локальна змінна.

Це важливо розуміти для запобігання неочікуваній поведінці програми.

Приклад

```
var variable = "Глобальна змінна";

function showVariable() {

let variable = "Локальна змінна";

console.log(variable);

}
```

Результат

showVariable(); // Виведе: Локальна змінна

console.log(variable); // Виведе: Глобальна змінна

Всередині функції використовується локальна змінна, а поза нею - глобальна.

Повернення значень з функцій

Оператор return

Функції можуть повертати значення за допомогою оператора return. Це значення можна використовувати поза межами функції.

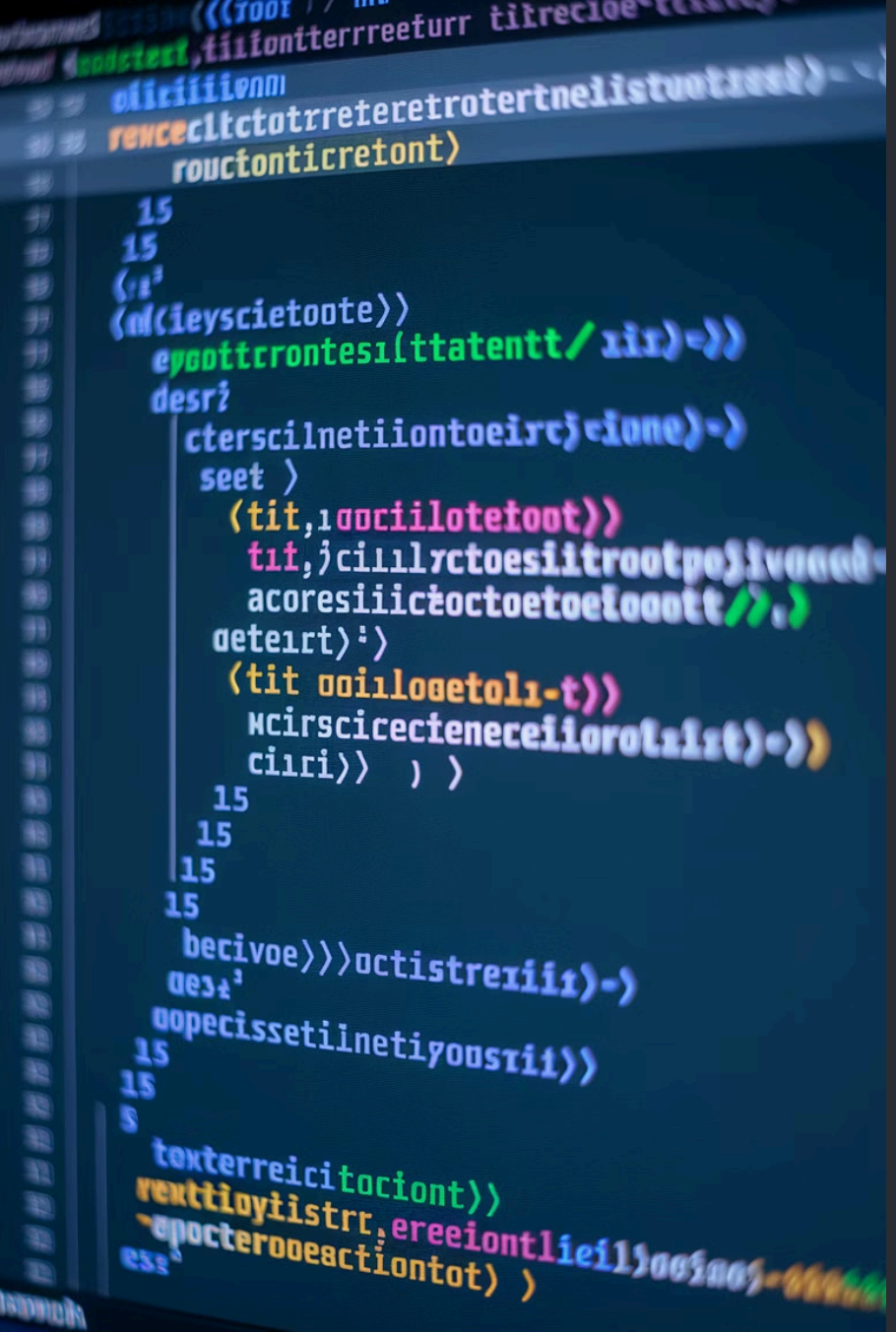
Приклад функції

```
function add(a, b) { return a + b; }
```

Використання результату

```
let result = add(5, 3);
console.log(result); // Виведе: 8
```

Повернення значень дозволяє функціям не лише виконувати дії, але й передавати результати своєї роботи для подальшого використання в програмі.



Функції без повернення значення

Значення за замовчуванням

Функції, які не містять оператора return, повертають undefined за замовчуванням. Це вказує на відсутність повернутого значення.

Приклад

```
function noReturn() {
console.log("Ця функція нічого не повертає"); }
```

Результат

```
let resultNoReturn = noReturn(); // Виведе: Ця функція нічого не повертає

console.log(resultNoReturn); // Виведе: undefined
```

Шаблон раннього виходу для функцій

1

Визначення

Шаблон раннього виходу дозволяє завершити виконання функції, якщо певна умова виконується. Це покращує продуктивність і робить код більш зрозумілим.

2

Приклад

```
function перевіритиВік(вік) {
if (вік < 0) { return "Невірний вік"; }
if (вік < 18) { return "Ви неповнолітні"; }
return "Ви повнолітні";
}
```

3

Результат

```
console.log(перевіритиВік(20)); // "Ви повнолітні"
```

Дебагінг у Chrome: Відкриття консолі

Крок 1

Натисніть правою кнопкою миші на веб-сторінці.

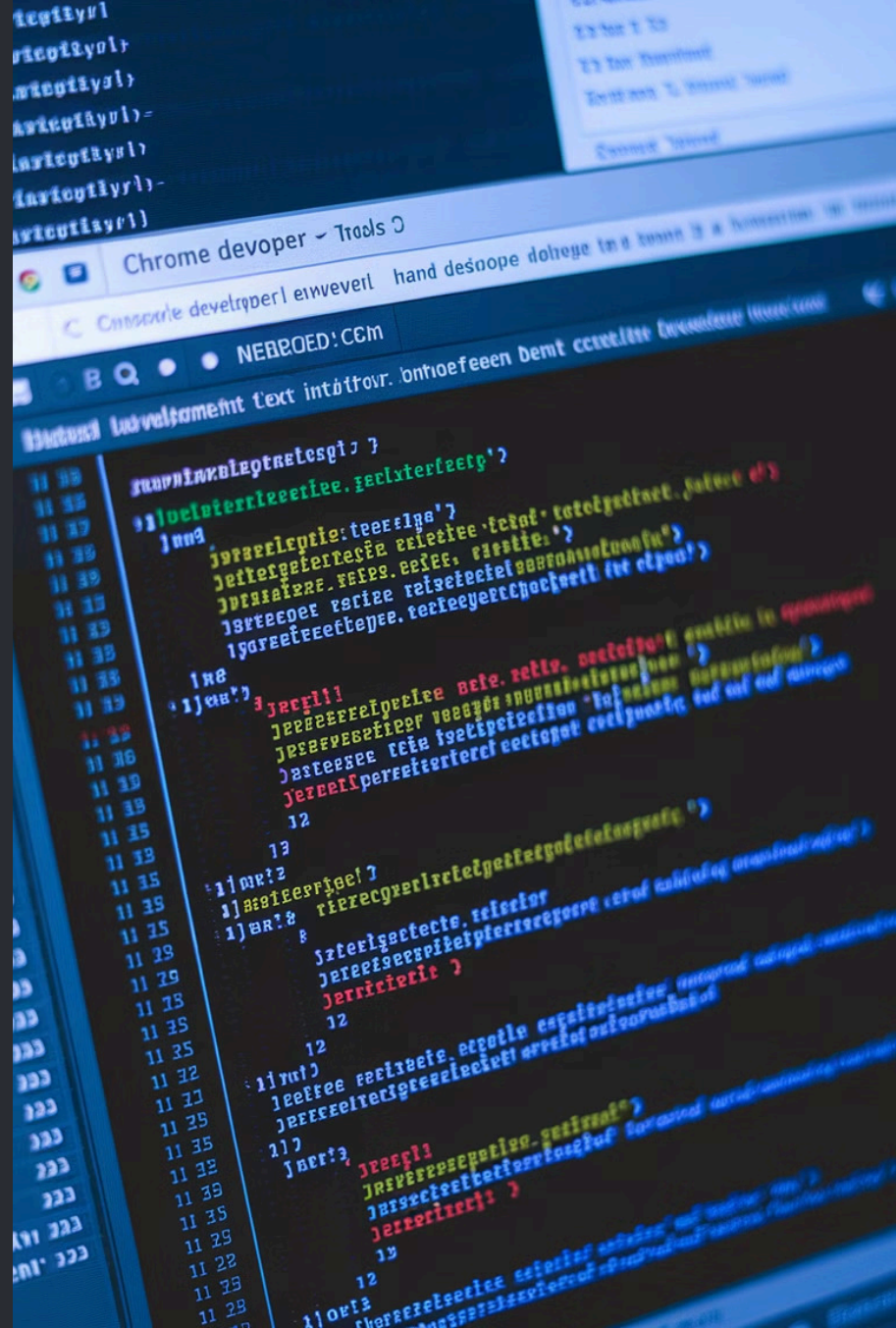
Крок 2

Виберіть "Переглянути код" або натисніть Ctrl+Shift+I (Windows/Linux) або Cmd+Option+I (Mac).

Крок 3

Перейдіть на вкладку "Console".

Консоль розробника - це потужний інструмент для відлагодження JavaScript коду. Вона дозволяє переглядати повідомлення про помилки, виводити значення змінних та тестувати фрагменти коду безпосередньо у браузері.



Дебагінг у Chrome: Вхід у дебагер

1

Крок 1

Відкрийте консоль, як описано раніше (Ctrl+Shift+I або Cmd+Option+I).

2

Крок 2

Перейдіть на вкладку "Sources".

3

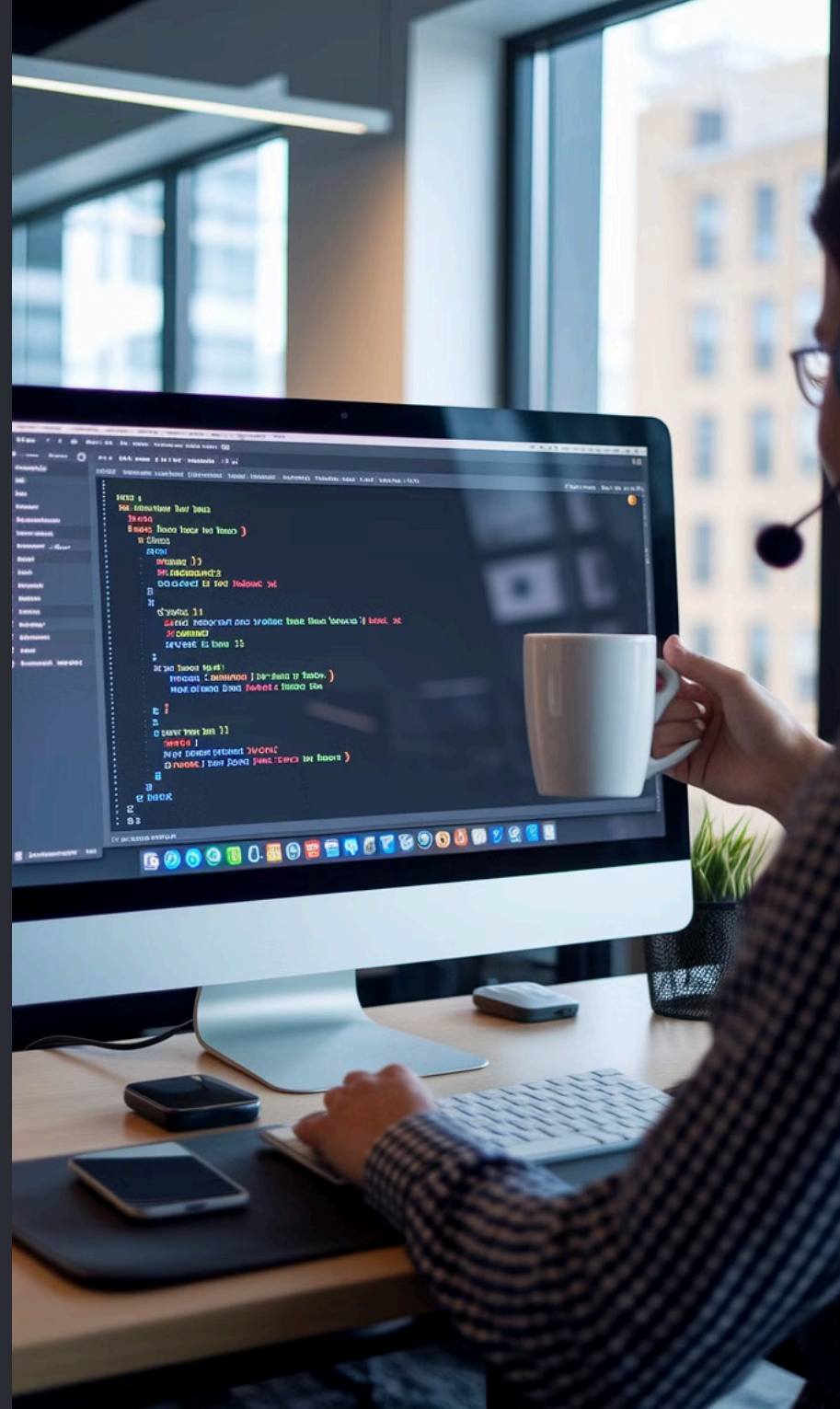
Крок 3

Виберіть файл JavaScript, який ви хочете відлагоджувати.

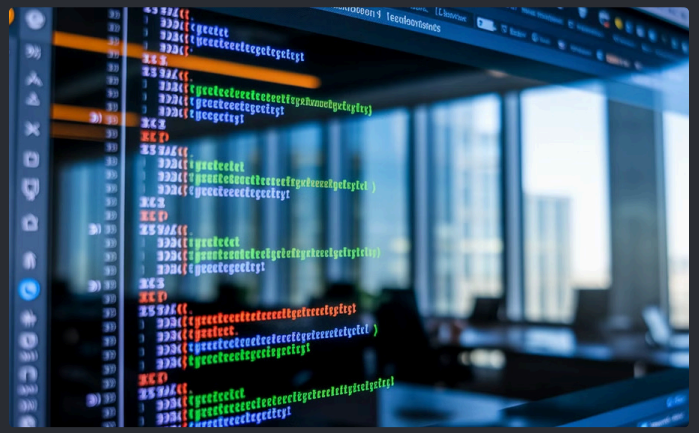
4

Крок 4

Натисніть на номер рядка, де ви хочете встановити точку зупину (breakpoint).



Дебагінг у Chrome: Watch вирази та брекпоінти

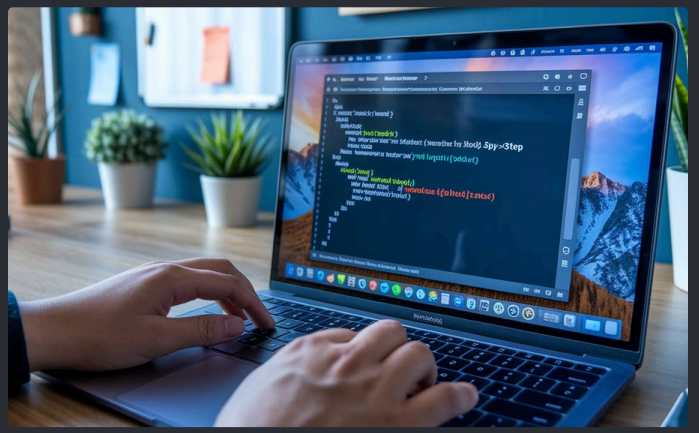


Watch вирази

Watch вирази дозволяють відстежувати значення змінних та виразів під час виконання коду. Вони особливо корисні для моніторингу змін у змінних під час покрокового виконання програми.

Брекпоінти

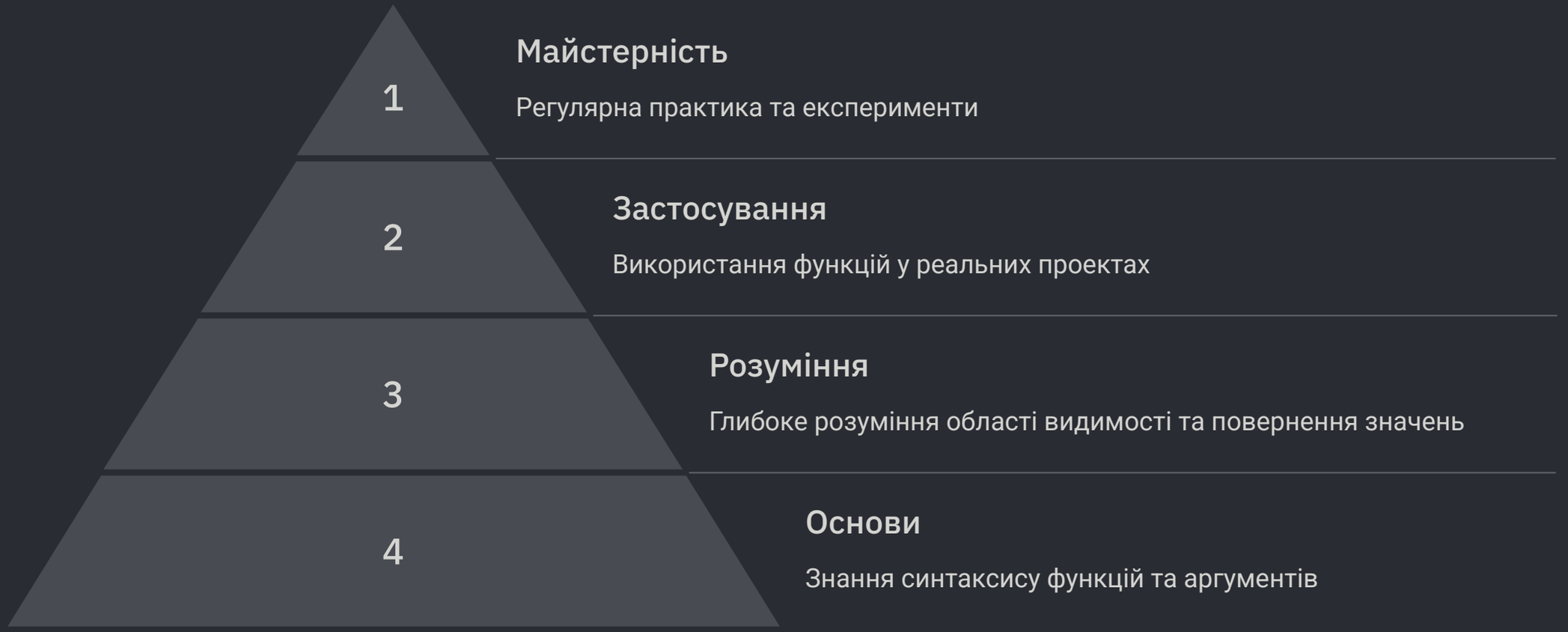
Брекпоінти (точки зупину) дозволяють призупинити виконання коду в певному місці. Це дає можливість перевірити стан програми в конкретний момент часу та виявити помилки.



Покрокове виконання

Після зупинки на брекпоінті можна виконувати код покроково, використовуючи кнопки Step Over, Step Into та Step Out. Це дозволяє детально проаналізувати логіку роботи програми.

Підсумки та практичні поради



У цій лекції ми розглянули функції, аргументи, параметри за замовчуванням, область видимості та повернення значень. Ці основні концепції допоможуть вам розпочати програмування на JavaScript та підготують вас до більш складних тем у майбутніх лекціях.

Пам'ятайте, що практика - ключ до успіху в програмуванні. Спробуйте створити власні функції, експериментуйте з різними типами аргументів та областями видимості, щоб закріпити отримані знання.