Практична робота №5

Тема: Оголошення та виклик функцій: від Declaration до Arrow Functions.

Мета: Відпрацювати синтаксичні відмінності між видами функцій, зрозуміти роботу параметрів за замовчуванням та повернення значень.

КОРОТКІ ТЕОРЕТИЧНІ ВІДОМОСТІ

Функції − це блоки коду повторного використання, призначені для виконання певного завдання. Функції виконуються, коли їх викликають. Функції є фундаментальними у всіх мовах програмування. Існує три способи оголошення функцій.

1. Класичний спосіб оголошення функції (Declaration).

Функції оголошуються з таким синтаксисом:

```
function functionName(parameters) {
  // code to be executed
}
```

Функцію можна створити за допомогою ключового слова `function` , імені та дужок. Код для запуску пишеться у фігурних дужках.

Приклад:

```
function getAreaDeclaration(width, height) {
    return width * height;
}
console.log(getAreaDeclaration(5, 10)); // 50
```

Примітка:

- Hoisting (Підняття). Головна особливість. Цю функцію можна викликати до того, як вона описана в коді. Браузер спочатку "сканує" файл на наявність таких оголошень і піднімає їх у пам'яті в саму гору.
- Синтаксис: Починається з ключового слова function, має ім'я та блок коду.

2. Функціональний вираз (Function Expression).

Тут функція розглядається як значення, яке ми записуємо у змінну.

```
const getAreaExpression = function(width, height) {
    return width * height;
};
console.log(getAreaExpression(5, 10)); // 50
```

Примітка:

- Відсутність Hoisting. Ви не можете викликати цю функцію вище рядка, де вона створена. Для програми це просто змінна, яка стає функцією тільки після виконання рядка з присвоєнням.

- Анонімність: Сама функція часто не має імені (анонімна), її ім'ям стає назва змінної. Це робить код більш структурованим, змушуючи розробника спочатку оголошувати інструменти, а потім ними користуватися.

3. Стрілкова функція (Arrow Function)

Сучасний стандарт (ES6), що з'явився у 2015 році. Це найбільш лаконічний спосіб.

```
const getAreaArrow = (width, height) => width * height;

console.log(getAreaArrow(5, 10)); // 50
```

Примітка:

- Лаконічність: Якщо функція має лише один вираз, можна прибрати фігурні дужки `{}` та ключове слово `return`. Повернення значення відбувається автоматично (неявне повернення).
- Відсутність `this`: У стрілкових функцій немає власного контексту `this`. Вони беруть його з оточення. Це робить їх ідеальними для методів масивів або обробників подій, але не для методів об'єктів.
- Hoisting: Як і вирази, вони не піднімаються.

Параметри (вхідні дані функції)

Параметри дозволяють передавати (надсилати) значення до функції. Параметри перелічені в дужках у визначенні функції. Ви можете додати скільки завгодно параметрів, розділяючи їх комами.

Наприклад:

```
function fullName(firstName, lastName) {
    return firstName + " " + lastName;
}

let name = fullName("John", "Doe");
console.log(name); // John Doe
```

Параметри проти аргументів

У JavaScript параметри та аргументи функцій є різними поняттями:

- ***Параметри*** – це імена, перелічені у визначенні функції.
- ***Аргументи*** – це реальні значення, що передаються функції та отримуються нею.

Параметри за замовчуванням

Якщо функція викликається з відсутніми аргументами (менше, ніж оголошено), відсутні значення встановлюються на undefined.

Іноді це прийнятно, але іноді краще призначити параметру значення за замовчуванням.

ECMAScript 2015 дозволяє параметрам функцій мати значення за замовчуванням. Тепер можна встановити значення за замовчуванням для параметра.

Значення за замовчуванням використовується, якщо не надано аргумент.

Приклад:

Якщо `y` не передано або не визначено, тоді `y = 10`.

```
function myFunction(x, y = 10) {
  return x + y;
}
myFunction(5);
```

Параметр залишку функції

Параметр rest (...) дозволяє функції обробляти невизначену кількість аргументів як масив.

Приклад:

```
function sum(...args) {
  let sum = 0;
  for (let arg of args) sum += arg;
  return sum;
}

let x = sum(4, 9, 16, 25, 29, 100, 66, 77);
```

ЗАВДАННЯ ДЛЯ САМОСТІЙНОГО ВИКОНАННЯ

Завдання 1. Синтаксичний практикум.

Написати одну й ту саму функцію обчислення площі круга трьома способами: Function Declaration, Function Expression та Arrow Function.

Завдання 2.Параметри за замовчуванням.

Створити функцію вітання sayHello (name, greeting = "Привіт"). Перевірити її роботу, передаючи обидва аргументи та лише один.

Завдання 3.Область видимості (Scope).

Створити глобальну змінну та локальну всередині функції. Продемонструвати в консолі, чому локальна змінна недоступна зовні.

Завдання 4.Функція-колбек (Callback):

Написати функцію processNumbers (a, b, callback), яка приймає два числа та функцію, що визначає, яку операцію з ними зробити (додати чи помножити).