**Лабораторная работа №5. Функции и замыкания**

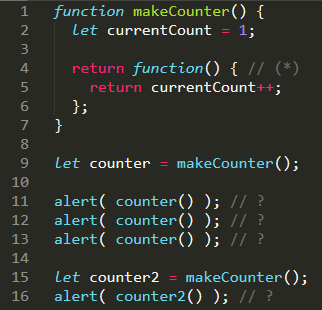
*Глобальный объект. Замыкания, внутренняя работа функции, лексическое окружение. Каррирование. Генераторы.*

**Задание:** изучите теорию и решите задачи.

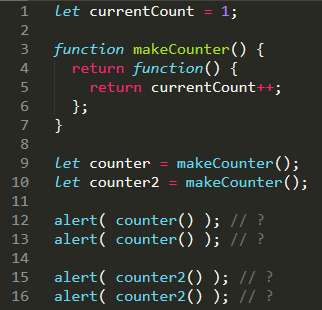
**Задачи:**

1. Что выведет alert в примерах? Поясните почему так? На что ссылается [[Environment]] функций? Что будет содержать LexicalEnvironment при запуске функций? Что хранится в counter? Когда будет вызвана функция (\*)?

Вариант 1.



Вариант 2.



В двух приведённых вариантах кода разная организация переменных влияет на результат работы замыкания. Давайте разберём каждый вариант отдельно:

### Вариант 1:

1. В функции `makeCounter()` создаётся локальная переменная `currentCount`, равная 1.

2. Эта переменная остаётся доступной через замыкание (функция, возвращаемая из `makeCounter()`).

3. Когда вызывается `counter()`, он ссылается на внутреннюю переменную `currentCount` функции `makeCounter` и увеличивает её на 1 после каждого вызова.

\*\*Результаты вывода alert:\*\*

- `alert(counter())` выведет 1 (переменная `currentCount` сначала равна 1, но потом увеличивается до 2).

- Следующий вызов `alert(counter())` выведет 2, так как `currentCount` увеличен в предыдущем вызове.

- Третий вызов выведет 3.

- `alert(counter2())` выведет 1, потому что `counter2` создаётся отдельным вызовом `makeCounter()`, в котором своя копия `currentCount`, равная 1.

\*\*На что ссылается `[[Environment]]` функции?\*\*

- Функция, возвращаемая `makeCounter()`, ссылается на лексическое окружение, где существует переменная `currentCount`.

\*\*Что будет содержать `LexicalEnvironment` при запуске функций?\*\*

- При каждом вызове функции `counter()` и `counter2()` `LexicalEnvironment` будет содержать переменную `currentCount`, которая хранит текущее значение счётчика.

\*\*Что хранится в `counter`?\*\*

- В переменной `counter` хранится ссылка на функцию, возвращённую `makeCounter()`. Эта функция содержит замыкание, которое захватывает значение `currentCount` и изменяет его при каждом вызове.

\*\*Когда будет вызвана функция (\*)?\*\*

- Функция (\*) будет вызвана каждый раз, когда вызывается `counter()` или `counter2()`.

### Вариант 2:

1. Переменная `currentCount` объявлена вне функции `makeCounter()` и глобальна для всех вызовов.

2. В результате все вызовы `counter()` и `counter2()` будут ссылаться на одну и ту же переменную `currentCount`.

\*\*Результаты вывода alert:\*\*

- Первый `alert(counter())` выведет 1.

- Второй вызов выведет 2, так как `currentCount` увеличилась после первого вызова.

- Третий вызов выведет 3.

- Первый вызов `counter2()` также выведет 4, так как обе функции `counter` и `counter2` ссылаются на одну и ту же глобальную переменную `currentCount`.

- Второй вызов `counter2()` выведет 5.

\*\*На что ссылается `[[Environment]]` функций?\*\*

- В данном случае функции `counter` и `counter2` будут ссылаться на одно и то же лексическое окружение, в котором существует глобальная переменная `currentCount`.

\*\*Что будет содержать `LexicalEnvironment` при запуске функций?\*\*

- В обоих случаях в `LexicalEnvironment` будет одна и та же глобальная переменная `currentCount`, и она будет увеличиваться при каждом вызове любой из функций.

\*\*Что хранится в `counter`?\*\*

- В переменной `counter` также хранится ссылка на замыкающую функцию, но она ссылается на глобальную переменную `currentCount`.

\*\*Когда будет вызвана функция (\*)?\*\*

- Функция (\*) будет вызвана при каждом вызове `counter()` и `counter2()`, но в этот раз они будут изменять одно и то же значение `currentCount`, так как оно глобально.

---

### Краткий вывод:

- В \*\*варианте 1\*\* каждая функция `counter` и `counter2` работает с независимой копией переменной `currentCount` из своего замыкания.

- В \*\*варианте 2\*\* обе функции используют одну и ту же глобальную переменную `currentCount`, что приводит к её общему изменению.

1. Реализуйте каррированную функцию, которая рассчитывает объем прямоугольного параллелепипеда. Выполните преобразование функции для неоднократного расчёта объема прямоугольных параллелепипедов, у которых одно ребро одинаковой длины.
2. Пользователь управляет движением объекта, вводя в модальное окно команды left, right, up, down. Объект совершает 10 шагов в заданном направлении (т.е. высчитываются и выводятся в консоль соответствующие координаты) и запрашивает новую команду. Разработайте генератор, который возвращает координаты объекта, согласно заданному направлению движения.
3. Какие переменные и функции сохраняются в глобальный объект window? Получите значения всех созданных переменных и функции, которые хранятся в глобальном объекте window. Переопределите переменные через глобальный объект.

Глобальные переменные и функции в JavaScript автоматически добавляются в объект window.