**Функции в Swift**

**План урока:**

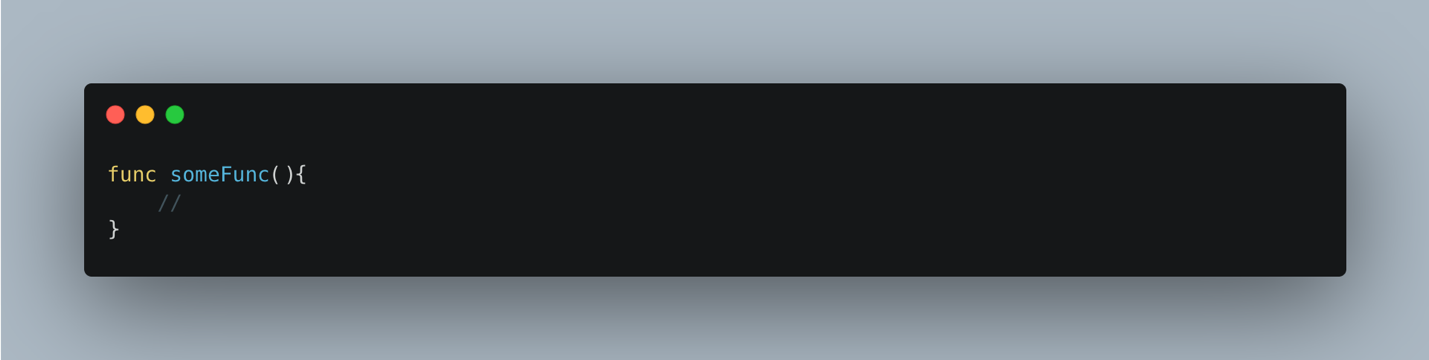
* Зачем нужны функции
* Особенности создания функций в Swift

Функции в Swift, как и в остальных языках программирования, являются очень важной и неотъемлемой частью. В функцию мы можем вынести код, который хотим выполнить несколько раз, или нужно выполнить по вызову. Так же в функции пишется код, который в зависимости от входящих параметров возвращает разный ответ.

В Swift существует несколько разновидностей функций, хотя по своему существу это одно и то же:

* Функция, которая ничего не принимает и ничего не возвращает
* Функция, которая принимает данные, но не возвращает ничего
* Функция принимает данные и возвращает ответ

Рассмотрим на примере простую функцию:

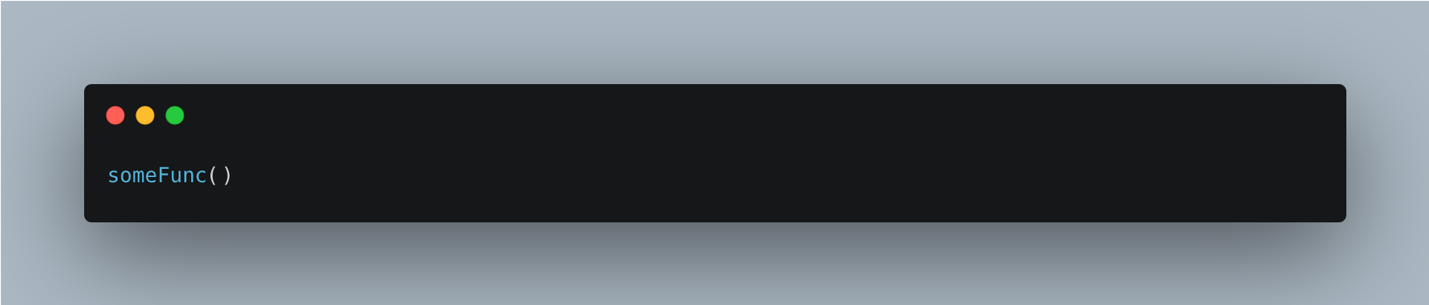


Синтаксис создания функции, следующий:

- Сначала пишем слово **func**

- Затем ставим скобки **()** – данном примере у нас функция, которая ничего не принимает и не возвращает, поэтому содержимое скобок оставляем пустыми.

- И затем идут фигурные скобки, это тело самой функции, здесь мы пишем код, который будет выполняться при вызове функции.

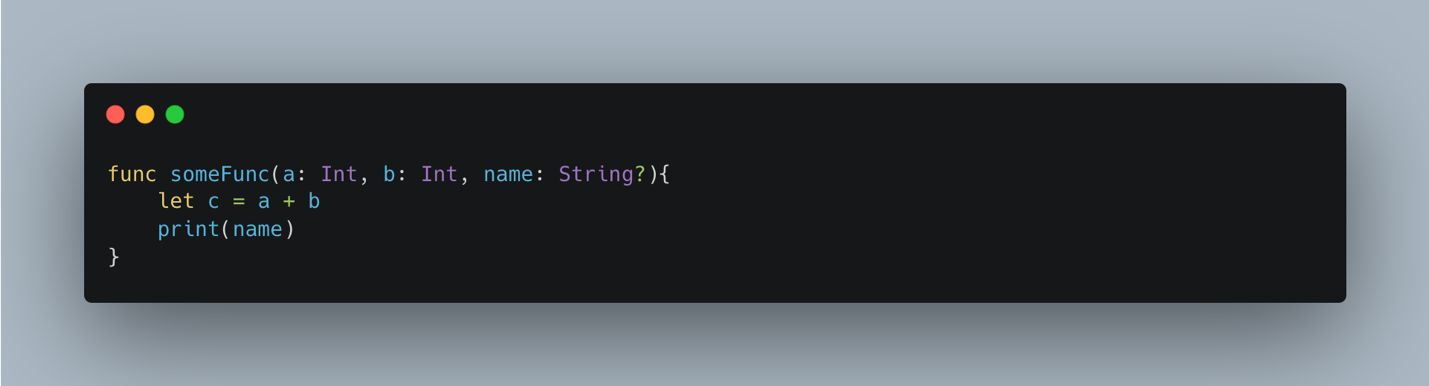


Чтобы вызвать такую функцию, достаточно всего лишь написать в нужном нам месте следующее:

В момент вызова будет выполняться код из функции.

Но допустим, нам нужно передать что-то в функцию, это частая практика, когда у нас функция принимает параметры, и в зависимости от них, выдает разный результат.

Пример функции с параметрами



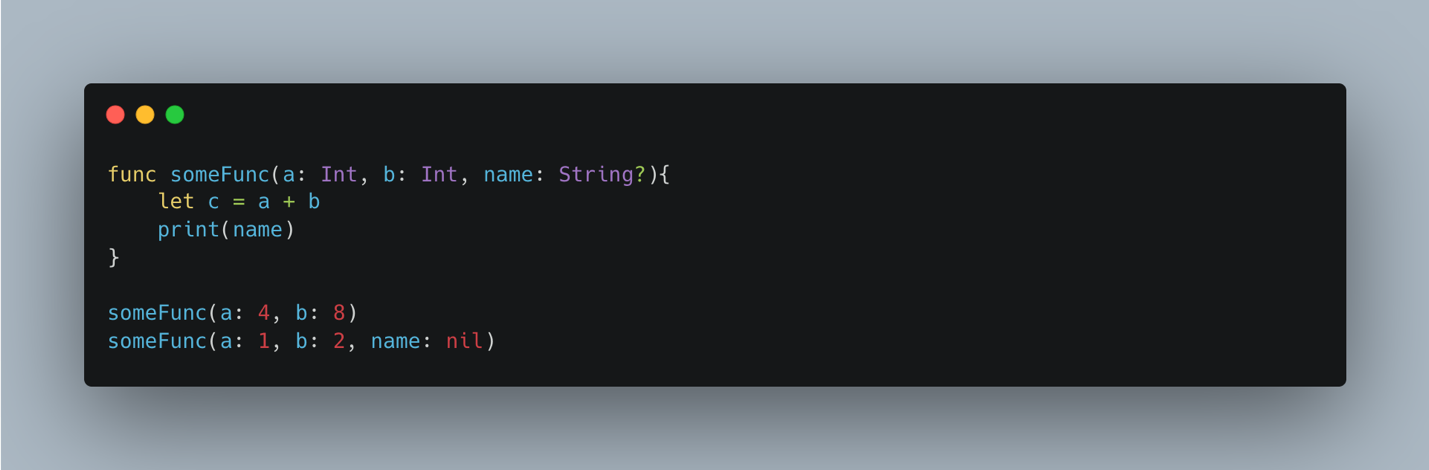
Здесь мы в скобках пишем названия параметров, под этими именами они будут доступны внутри тела функции, и тип входящих параметров. Можно сказать, что тут у нас создаются константы (их нельзя изменить внутри функции) и присваиваются им тип, типы могут быть опциональными. Далее внутри функции мы можем совершать с ними любые операции. Важно, эти переменные доступны только внутри функции, за ее пределами их нельзя вызвать.

Во время вызова такой функции, мы должны передать в нее обязательные аргументы:

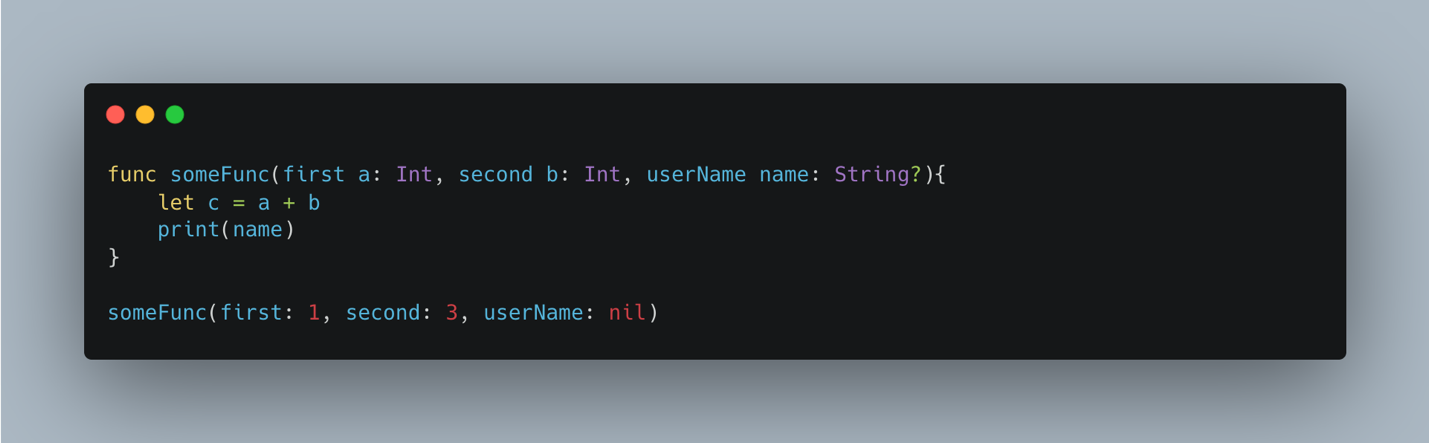


Важно: Тип передаваемых аргументов должен соответствовать типу принимаемых аргументов внутри функции, иначе будет ошибка и ничего не заработает.

Если внутри функции опциональный аргумент, то ее мы можем не передавать либо передать nil



Допустим мы хотим внутри фунции использовать названия переменных, отличных от входящих, к примеру, сейчас функция принимает параметр **a** и внутри функции мы так же используем ее по названию **a**. Крайне редко, но бывает такие ситуации, когда необходимо принимать параметр под одним названием, а использовать ее под другим, поэтому мы можем переписать нашу функцию следющим образом:



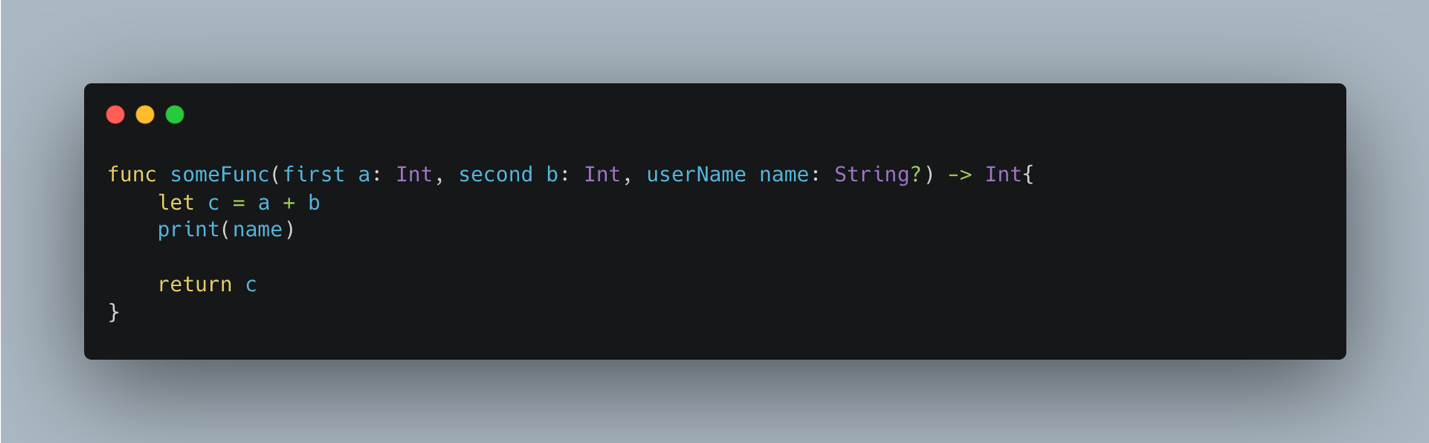
Теперь мы в аргументах функции пишем следующее (Важно, все названия просто из головы, они не унифицированы и вы можете задавать какие вам удобно):

first a: Int – название **first** – будет использоваться при вызове фунции, а в теле функции будем использовать имя **a**

И соответственно у нас меняется сам вызов функции.

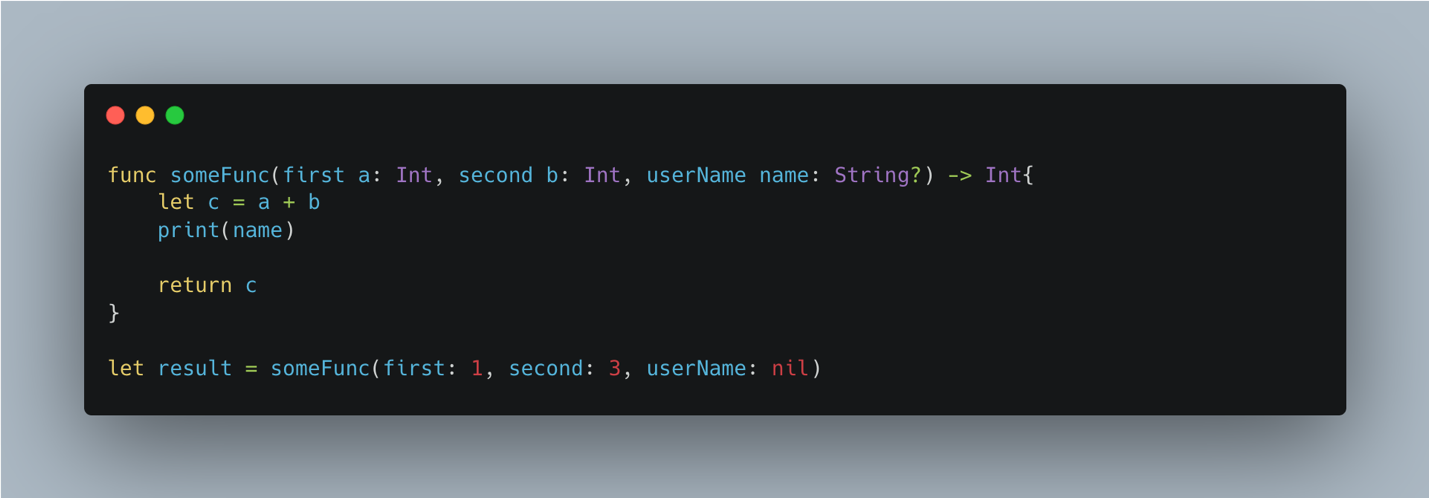
И последний пример, это функция которая может вернуть нам какой – то ответ. Предыдущие функции у нас выполняли некий код, но никакого ответа от них мы не получали, а представьте, что нам надо вернуть некое значение, или результат выполнения функции.

Синтаксис такой функции выглядит следующим образом:



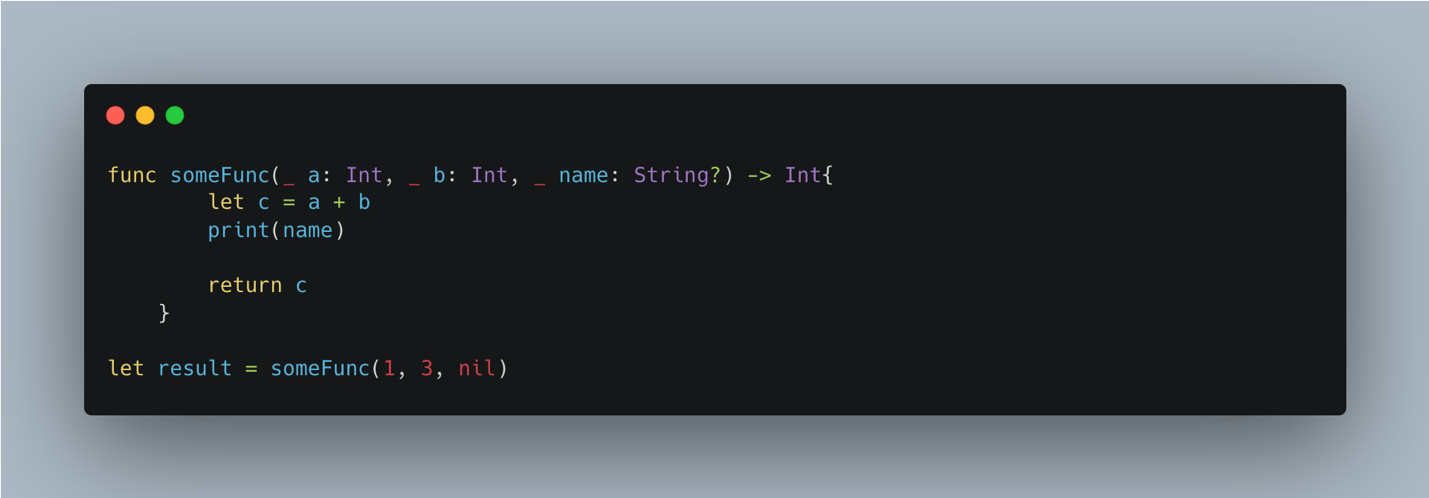
Здесь, после скобок мы пишем следующее -> и тип возвращаемого ответа. Тип может быть любой, даже опциональным. И если мы говорим, что ждем от функции ответ, то обязательно в теле функции должно быть return и указываем возвращаемое значение. Важно, тип возвращаемого значения должен совпадать с указанным при создании функции, так же нужно понимать, что после return код никакой не выполняется, это фактически означает, что мы выходим из функции, поэтому писать после слова return код бесмысленно.

Вызов такой функции ничем не отличается от предыдущих, только теперь результат мы можем присвоить переменной:



И теперь переменная result присвоиться значение, которое вернет функция, в данном случае 4 и тип int.

Допустим, что у нас функция принимает много значений, и мы хотим немного упростить вызов, к примеру, не писать во время вызова названия аргументов, а сразу передавать туда значения, для этого можно написать следующее:



Перед аргументами в функции мы добавили знак \_ и после название переменной, теперь, при вызове функции, мы можем опустить названия аргументов и сразу передать нужные значения, это немного упрощает наш код.

ДЗ.

1. Создать функцию, которая принимать массив из int и возвращать новый массив, который будет содержать только четные/нечетные числа
2. Написать простой калькулятор – функция принимает два аргумента float и принимает строку, символ операции (+-\*/) и в зависимости команды, сделать математическую операцию с цифрами и вернуть результат

**Доп. Ссылки**

<https://docs.swift.org/swift-book/LanguageGuide/Functions.html>

<https://docs.swift.org/swift-book/LanguageGuide/StringsAndCharacters.html>