Trailing closure syntax in Swift allows you to write closures as arguments to functions, right after the function call, inside parentheses. This can be a more concise and expressive way to write closures, especially when the closure is the last argument to a function.

For example, the following code shows how to use trailing closure syntax to pass a closure to the `map()` function:

```
Swift
let numbers = [1, 2, 3, 4, 5]

let doubledNumbers = numbers.map { $0 * 2 }

print(doubledNumbers) // [2, 4, 6, 8, 10]
```

In this example, the closure that is passed to the `map()` function takes a single argument, which is a number. The closure then returns the number multiplied by 2.

Without trailing closure syntax, the same code would look like this:

```
Swift
let numbers = [1, 2, 3, 4, 5]

let doubledNumbers = numbers.map { (number: Int) -> Int in
    return number * 2
}

print(doubledNumbers) // [2, 4, 6, 8, 10]
```

As you can see, the trailing closure syntax is more concise and expressive. It also makes the code more readable, because it is clear that the closure is an argument to the `map()` function.

Empty parentheses of a function, (), can be deleted when using trailing closure syntax. This is because the parentheses are no longer needed to indicate that the closure is an argument to the function.

For example, the following code is equivalent to the previous example:

```
Swift
let numbers = [1, 2, 3, 4, 5]
```

```swift
let doubledNumbers = numbers.map { $0 * 2 }

print(doubledNumbers) // [2, 4, 6, 8, 10]
```

In general, it is a good practice to use trailing closure syntax and delete empty parentheses of a function when possible. This will make your code more concise, expressive, and readable.

Here are some additional tips for using trailing closure syntax in Swift:

- Trailing closure syntax can only be used when the closure is the last argument to a function.
- Trailing closure syntax can be used with both optional and non-optional closures.
- Trailing closure syntax can be used with both closures that take arguments and closures that do not take arguments.

Overall, trailing closure syntax is a valuable tool for any Swift developer. It allows you to write more concise, expressive, and readable code.