Информация по append в Go

`append` — одна из самых важных встроенных функций в Go для работы со слайсами. Ниже рассмотрены все возможные способы использования, примеры и пояснения.

□□ Базовый синтаксис

```
append(slice, elems...)
```

- `slice` — срез, к которому добавляем
- `elems...` — один или несколько элементов **или другой срез**

□ Варианты использования

1. Добавить один элемент

```
nums := []int{1, 2}
nums = append(nums, 3)
// → [1 2 3]
```

---

2. Добавить несколько элементов

```
nums = append(nums, 4, 5, 6)
// → [1 2 3 4 5 6]
```

---

3. Добавить другой срез

```
more := []int{7, 8}
nums = append(nums, more...) // обязательно `...`!
```

---

4. Добавить пустой срез

```
nums = append(nums, []int{}...) // всё ещё [1 2 3 4 5 6 7 8]
```

---

5. Добавить к `nil`-срезу

```
var s []int
s = append(s, 10) // работает, даже если s == nil
```

---

6. Удалить элемент (по индексу)

```
i := 2
s = append(s[:i], s[i+1:]...)
```

□□ Как работает `append`

- Если у слайса **есть свободная capacity** — элементы просто добавляются.
- Если нет — Go **создаёт новый массив**, копирует старые значения и добавляет новые.
- Поэтому `append` может **изменить underlying array**.

□□ Трюки с `append`

Объединение слайсов

```
a := []string{"a", "b"}
b := []string{"c", "d"}
a = append(a, b...) // → [a b c d]
```

Удаление последнего элемента

```
s = s[:len(s)-1]
```

⚠ Важно помнить

- `append` **всегда возвращает новый слайс** (возможно тот же, возможно новый)
- Обязательно `...`, если добавляешь другой срез: `append(s, other...)`
- Работает **только со слайсами**, не с массивами

Сводка

Использование	Пример
Один элемент	`append(s, 10)`
Несколько элементов	`append(s, 1, 2, 3)`
Добавить срез	`append(s, other...)`
Удалить элемент	`append(s[:i], s[i+1:]...)`
Работать с `nil`-срезом	`append(nil, 1)`

Revision #1
Created 2025-06-20 11:50:14 UTC by Admin
Updated 2025-06-20 11:51:08 UTC by Admin