Конспект «Циклы». Раздел 1

Цикл for

Синтаксис

```
for (let i = 0; i < 10; i++) {

  // Повторяющиеся команды

}
```

В круглых скобках записывается код управления циклом. Он состоит из трёх частей, разделённых `;`.

1. Первая часть — подготовительная. Команды отсюда запускаются *один раз* перед началом работы цикла. Обычно здесь задаётся исходное значение для переменной-счётчика. Обратите внимание, что в цикле мы создаём переменную-счётчик с помощью `let`, как в случае с любой другой переменной.

```
for (let i = 0; i < 5; i = i + 1) { }
```

2. Вторая часть — проверочная. Она содержит условие и запускается *перед* каждым новым витком цикла. Если условие возвращает `true`, цикл делает ещё один виток, иначе цикл завершает свою работу.

```
for (let i = 0; i < 5; i = i + 1) { }
```

3. Третья часть — дополняющая, или «закон изменения». Код третьей части запускается *после* каждого витка цикла. Обычно там изменяется переменная-счётчик.

```
for (let i = 0; i < 5; i = i + 1) { }
```

Накопление значений в цикле:

Внутри циклов можно использовать обычные математические операции. Например, сложение:

```
let sum = 0;



for (let i = 1; i <= 5; i++) {

  sum += 2;

  console.log(sum);
```

```
}
```

Программа выведет:

```
LOG: 2 (number)

LOG: 4 (number)

LOG: 6 (number)

LOG: 8 (number)

LOG: 10 (number)
```

Проверки в теле цикла

Если добавить условие внутрь цикла, то оно будет проверяться на каждой итерации.

```
let sum = 0;



for (let i = 1; i <= 5; i++) {

  if (i > 2) {

    sum += 1;

  }

}
```

Поиск чётного числа

Оператор `%`, или «остаток от деления», возвращает остаток от деления.

```
10 % 5;  // Вернёт 0

12 % 5;  // Вернёт 2

7 % 3;   // Вернёт 1

5.5 % 2; // Вернёт 1.5
```

Если остаток от деления числа на `2` равен `0` — число чётное, иначе нечётное.

Сокращённые операторы

В JavaScript есть несколько удобных операторов, которые позволяют сократить код:

Название	Пример	Ан
Инкремент (увеличение на единицу)	`i++`	`i`
Декремент (уменьшение на единицу)	`i--`	`i`
К-к-комбо!	`i += 2`	`i`

Комбинировать можно не только сложение, но и остальные математические операции: вычитание `-=`, умножение `*=`, деление `/=` и нахождение остатка `%=`.

Конспект «Циклы». Раздел 2

Цикл while

Синтаксис

```
while (условие) {

  действия

}
```

Действия будут выполняться снова и снова, пока условие не вернёт `false`.

Чтобы цикл остановился, условие когда-нибудь должно стать ложным. Если условие выхода из цикла не срабатывает, то цикл не может остановиться. Это бесконечный цикл, одна из любимых ошибок программистов.

break и continue

Оператор `break` прерывает выполнение цикла.

Аналогично оператору прерывания цикла `break` существует оператор для быстрого перехода к следующей итерации цикла `continue`, но используют его крайне редко, так как он усложняет чтение кода и понимание работы цикла в целом. Использование `continue` без необходимости обычно является дурным тоном.

- Внутри `while` команда `continue` «перематывает» программу сразу к началу *следующей* итерации.

- Внутри `for` команда `continue` «перематывает» программу к дополнительной части *текущей* итерации, после выполнения которой начинается *следующая* итерация цикла.

Накопление значений в цикле

```
let sum = 0;

let i = 0;



while (i <= 5) {

  sum += 1;

  i++;

  console.log(i);

}
```

Программа выведет:

```
LOG: 1 (number)

LOG: 2 (number)

LOG: 3 (number)

LOG: 4 (number)

LOG: 5 (number)

LOG: 6 (number) // Код из тела цикла не выполнится, условие вернёт false
```

Поиск процента от числа

Самый простой способ найти процент от числа — разделить число на 100 и умножить на процент.

```
// Найдём 2 процента от 1000

1000 / 100 * 2 = 20;



// Найдём 7 процентов от 1200

1200 / 100 * 7 = 84;
```