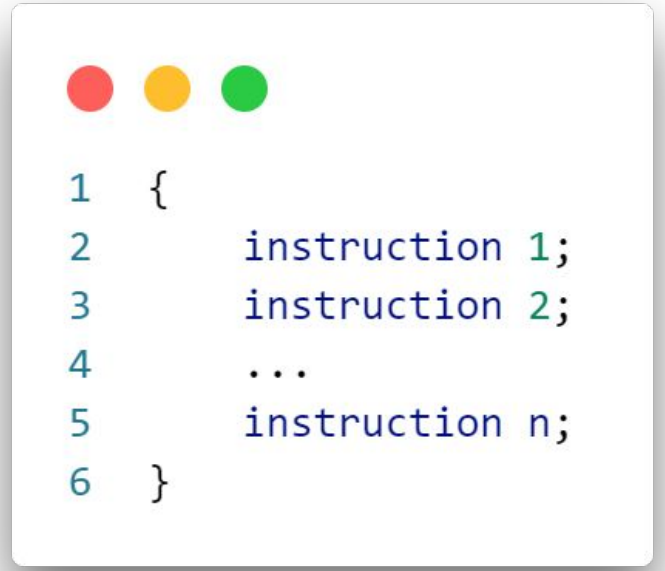
Алгоритмы обработки данных на JavaScript

Блоки и правила видимости переменных
Условия, циклы

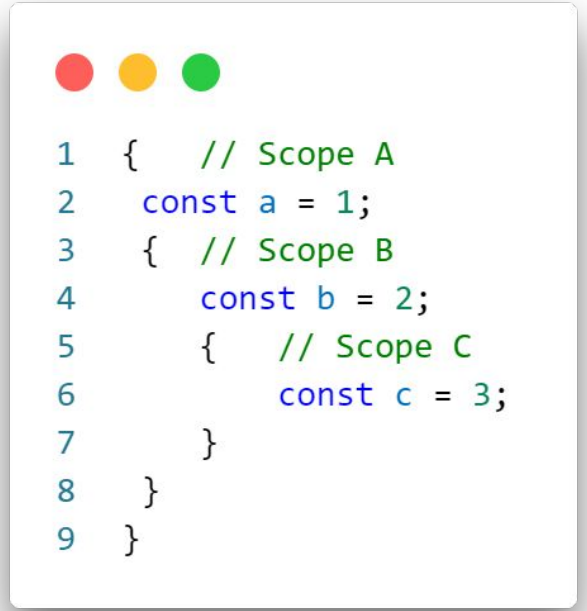
Блок



```
1  {
2      instruction 1;
3      instruction 2;
4      ...
5      instruction n;
6  }
```

Блок инструкций используется для группировки нуля или более инструкций. Блок отделяется парой фигурных скобок.

Область видимости переменной

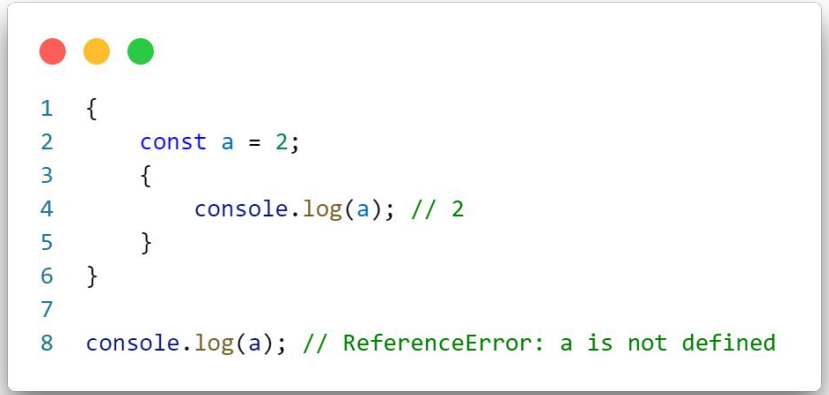


```
1  {   // Scope A
2   const a = 1;
3   {  // Scope B
4      const b = 2;
5      {   // Scope C
6          const c = 3;
7      }
8   }
9  }
```

Область видимости переменной - это часть программы в которой она доступна

- Scope A - область видимости переменной a
- Scope B - область видимости переменных a, b
- Scope C - область видимости переменных a, b, c

Область видимости переменных

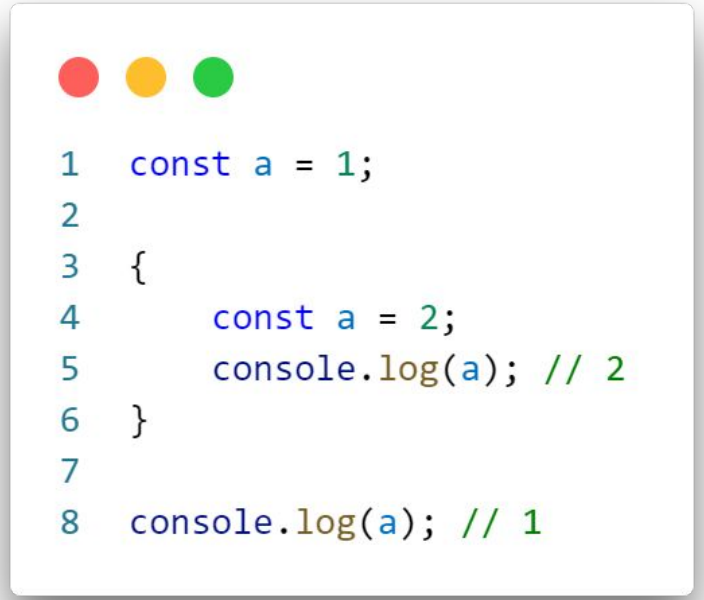


```
{
    const a = 2;
    {
        console.log(a); // 2
    }
}

console.log(a); // ReferenceError: a is not defined
```

Переменные, объявленные при помощи let и const, имеют блочную области видимости.

Не видны за пределами своего (и вложенных) блоков

Затенение переменной (Variable shadowing)



```
1  const a = 1;
2
3  {
4      const a = 2;
5      console.log(a); // 2
6  }
7
8  console.log(a); // 1
```

Переменную можно переопределить во внутреннем блоке

В таком случае во внутренних блоках невозможно будет получить доступ к переменной, объявленной во внешнем окружении

Temporal Dead Zone

```
{
    console.log(a); // TDZ
    a + 1; // TDZ
    a++; // TDZ
    // TDZ
    let a = 23;
}
```

TDZ - термин, описывающий состояние переменной с начала её области видимости до объявления.

Обращение к переменной в TDZ выбрасывает ReferenceError

```
let a = 'outer';

{
    console.log(a);

    let a = 'inner';
}
```

Условные выражения

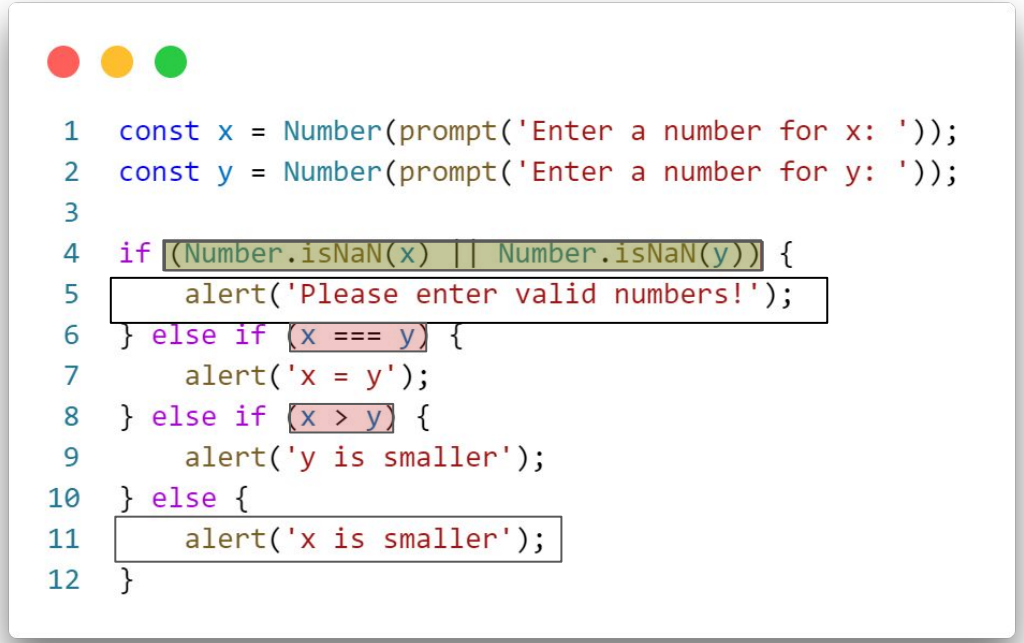
```
if (condition) {
    expresssion1;
    expresssion2;
    ...
}
```

```
if (condition) {
    expresssion1;
    expresssion2;
    ...
} else if (condition2) {
    expresssion3;
    expresssion4;
    ...
} else {
    expresssion5;
    ...
}
```

```
if (condition) {
    expresssion1;
    expresssion2;
    ...
} else {
    expresssion3;
    expresssion4;
    ...
}
```

- Условие - набор правил, который может прерывать нормальное выполнение кода или изменять его в зависимости от того, удовлетворено условие или нет
- condition приводится к булеву типу

Условные выражения



```
1  const x = Number(prompt('Enter a number for x: '));
2  const y = Number(prompt('Enter a number for y: '));
3
4  if (Number.isNaN(x) || Number.isNaN(y)) {
5      alert('Please enter valid numbers!');
6  } else if (x === y) {
7      alert('x = y');
8  } else if (x > y) {
9      alert('y is smaller');
10 } else {
11     alert('x is smaller');
12 }
```

Тернарный оператор

const result = условие ? выражение1 : выражение2;

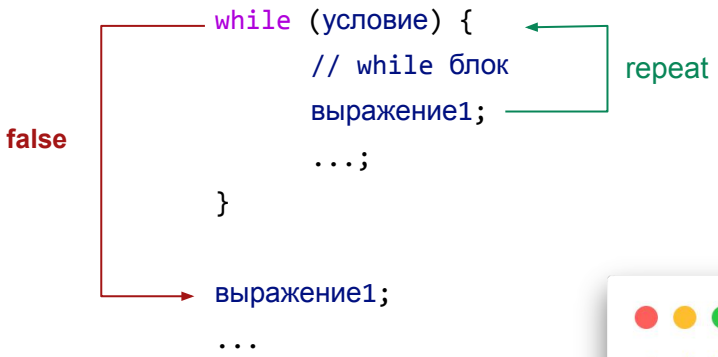
Оператор возвращает значение выражение1, если условие верно, и значение выражение2 в противном случае

```
1  const x = Number(prompt("Введите число: "));
2
3  const sign = x > 0 ? 1 : -1;
4
5  alert(`sign(${x}) = ${sign}`);
```

=

```
1  const x = Number(prompt("Введите число: "));
2
3  let sign;
4
5  if (x > 0) {
6      sign = 1;
7  } else {
8      sign = -1;
9  }
10
11 alert(`sign(${x}) = ${sign}`);
12
```

Циклы. while



- Если условие true, то выполняется while блок
- Снова проверяется условие
- Повторяем пока условие true

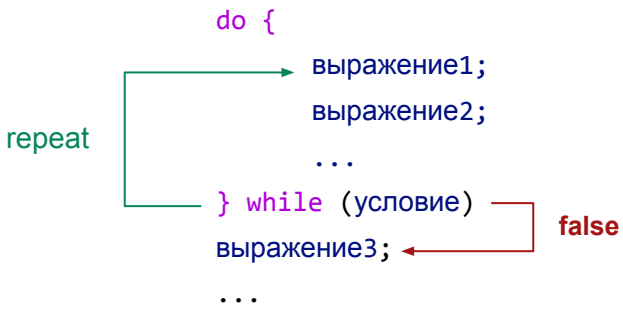
```
let sum = 0;
let n = 0;
const limit = 3;

while (n < limit) {
    sum += n;
    console.log(n);
    n++;
}

console.log('Сумма: ' + sum);

```

Шаг алгоритма	Операция	Переменные			Условие
		sum	n	limit	n < limit

1	let sum = 0;
	let n = 1;
	const limit = 3;
		0	1	3	

2	n < limit	0	1	3	1 < 3 (да)
3	sum += n	1	1	3	
4	n++	1	2	3	
5	n < limit	1	2	3	2 < 3 (да)
6	sum += n	3	2	3	
7	n++	3	3	3	
8	n <= limit	3	3	3	3 < 3 (нет)
9	console.log(‘Сумма’ + sum)	3	3	3	

Циклы. do…while

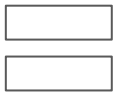


- Выполняется do блок
- Если условие true, то снова выполняется do блок
- Проверяется условие
- Повторяем пока условие true

```
let userName;

do {
    userName = prompt('Введите ваше имя');
} while (!userName);

alert(`Добрый день, ${userName}`);
```

Циклы. for

```
for (инициализация; условие; шаг) {
    тело;
}
```



```
инициализация;
while (условие) {
    тело;
    шаг;
}
```

```
1  for (let i = 0; i < 3; i++) {
2      console.log(i); //0,1,2
3  }
```

- инициализация - выполняется один раз при входе в цикл;
- условие - проверяется перед каждой итерацией цикла;
- шаг - выполняется после тела цикла перед проверкой условия

Шаг алгоритма	Операция	Переменные				Условие
		prod	a	pow	i	i < pow

1	let prod = 1;
	const a = 3;
	const pow = 3;
		1	3	3		

2	let i = 0;	1	1	3	0	
3	i < pow	1	1	3	0	0 < 3 (да)
4	prod *= a	3	2	3	0	
5	i++	3	2	3	1	
6	i < pow	3	2	3	1	1 < 3 (да)
7	prod *= a;	9	3	3	1	
8	i++	9	3	3	2	
9	prod * = a	27	3	3	2	
10	i++	27	3	3	3	
11	i < pow	27	3	3	3	3 < 3 (нет)

Оператор break

```
for (let x = 22; x < 40; x = x + 1) {
    if (x % 7 === 0) {
      console.log(x);
      break;
    }
}
```

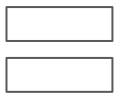
Оператор [break](#) прерывает выполнение текущего цикла

```
let sum = 0;

while (true) {
    const text = prompt('Введите q для выхода либо число');

    if (!text || text === 'q') {
        break;
    }

    const term = Number(text);

    if (!Number.isNaN(term)) {
        sum += term;
    }
}

alert('Сумма введённых чисел: ' + sum);
```

Оператор continue

```
for (let i = 0; i < 10; i++) {
    if (i % 2 === 0) {
        continue;
    }

    console.log(i);
}
```

Инструкция continue прерывает выполнение текущей итерации текущего цикла, и продолжает его выполнение для следующего шага



```
for (let i = 0; i < 10; i++) {
    if (i % 2 !== 0) {
        console.log(i);
    }
}
```

switch

```
let dayOfWeek;
let numberOfDay = 2;

switch (numberOfDay) {
    case 1: { // if (numberOfDay === 1)
        dayOfWeek = 'Понедельник';
        break;
    }
    case 2: { // else if (numberOfDay === 2)
        dayOfWeek = 'Вторник';
        break;
    }
    case 3: { // else if (numberOfDay === 3)
        dayOfWeek = 'Среда';
        break;
    }
    case 4: { // else if (numberOfDay === 4)
        dayOfWeek = 'Четверг';
        break;
    }
    case 5: { // else if (numberOfDay === 5)
        dayOfWeek = 'Пятница';
        break;
    }
    case 6: { // else if (numberOfDay === 6)
        dayOfWeek = 'Суббота';
        break;
    }
    case 7: { // else if (numberOfDay === 7)
        dayOfWeek = 'Воскресенье';
        break;
    }
    default: { // else
        dayOfWeek = '?';
    }
}
```

Конструкция switch сравнивает выражение со случаями, перечисленными внутри неё, а затем выполняет соответствующие инструкции

Используется как альтернатива if, ветви которого зависят от значения одной переменной

Если case блок не заканчивается break, то выполнение пойдёт ниже по следующим case, при этом остальные проверки игнорируются