Конспект «Функции». Раздел 1

Функция

Функция — кусок кода, который можно написать один раз, а затем многократно использовать. Функция не просто содержит в себе значение, как переменная, а выполняет какое-то действие и решает какую-то задачу: считает, сравнивает, ищет.

Код внутри `{ }` называется «телом функции».

```
let functionName = function () {

  // Тело функции

};
```

Чтобы функция начала свою работу, её надо вызвать. Для этого нужно обратиться к функции по её имени, а затем указать круглые скобки.

```
functionName();
```

Параметры и аргументы функции

Параметры — значения, с помощью которых можно настраивать функции. Так мы можем узнать результат работы функции для разных случаев.

В момент объявления функции, в круглых скобках, мы создаём параметры. Здесь всё, как с переменными: сначала задаём параметрам имена, которые описывают, что за значения будут в них записаны. Если параметров несколько, они записываются через запятую.

Параметры работают так же, как переменные. Мы подставляем их вместо фиксированных значений в операции внутри функции. При выполнении кода вместо каждого параметра подставится его значение.

В момент вызова функции мы указываем в круглых скобках те значения, которые окажутся в параметрах.

```
let showTime = function (hours, minutes) {

  console.log('Текущее время: ' + hours + ':' + minutes);

};



showTime(3, 15);    // Выведет: Текущее время: 3:15
```

```
showTime(16, 20);   // Выведет: Текущее время: 16:20
```

Правильно говорить «функция принимает параметры», но при этом мы «передаём функции **аргументы**».

Если у функции указан параметр, но аргумент не передан, то значение параметра в теле функции будет `undefined` — то есть «не определено».

Передавать аргументы надо **в том же порядке**, в котором объявлены параметры функции.

Возвращение из функции

Функции умеют **возвращать** результат своей работы. Это значит, что функция может выполнить код и отдать результат операций для дальнейшей работы с этим результатом. Он подставится в то место кода, где мы вызвали функцию.

Чтобы функция вернула значение, мы используем оператор `return`. После оператора указываем, что именно надо вернуть. Когда программа доходит до строки с `return`, функция отдаёт результат своей работы и выполнение кода из тела функции останавливается, иными словами *происходит выход из функции*.

- Код, написанный **на новой строке** после `return`, не выполняется.
- Функция не может вернуть сразу много значений, она возвращает **только один** результат.
- Если внутри функции нет `return` или после `return` не указано, какое значение нужно вернуть, функция вернёт `undefined`, иными словами, **ничего**.

Пример функции:

```
let calculateSum = function (numberFirst, numberSecond) {

  let sum = numberFirst + numberSecond;

  return sum;

};



calculateSum(); // Вернёт NaN

calculateSum(2); // Вернёт NaN

calculateSum(2, 5); // Вернёт 7

calculateSum(9, 5); // Вернёт 14
```

В этом примере:

- `calculateSum` — имя, по которому можно обратиться к функции.
- `numberFirst`, `numberSecond` — параметры функции.
- `return sum;` — место кода, где происходит возвращение `sum` и выход из функции.
- `calculateSum(2, 5);` — аргументы, которые передаются в функции при вызове. Порядок аргументов такой же, как у параметров функции. Первый аргумент `2` записывается в первый параметр `numberFirst`, аргумент `5` записывается в параметр `numberSecond`. Важно соблюдать порядок параметров при вызове функции, чтобы избежать неочевидных ошибок.

Конспект «Функции». Раздел 2

```
// Функция подсчёта миль



let calculateMiles = function (distance, isBusinessClass) {

  let percent = 0.18;

  if (isBusinessClass) {

    percent += 0.04;

  }

  if (distance > 3500) {

    percent += 0.15;

  }

  return distance * percent;

};




// Функция, которая считает количество полётов



let calculateFlights = function (distance, isBusinessClass, milesTarget) {

  // Вызываем одну функцию из другой

  let miles = calculateMiles(distance, isBusinessClass);

  let flights = Math.ceil(milesTarget / miles);

  return flights;

};
```

```
// Массив миль, которые нужно накопить



let targets = [1500, 3000, 5000, 7500, 10000, 15000];




// Цикл, в котором выясняется, какими перелётами мили накопятся быстрей



for (let i = 0; i < targets.length; i++) {

  let flightsVariantFirst = calculateFlights(3118, true, targets[i]);

  let flightsVariantSecond = calculateFlights(3617, false, targets[i]);



  console.log('Необходимое количество полётов в бизнес-классе до Валенсии: ' +
flightsVariantFirst);

  console.log('Необходимое количество полётов в экономе до Лиссабона: ' + flightsVariantSecond);



  if (flightsVariantFirst > flightsVariantSecond) {

    console.log('Быстрей накопишь полётами в экономе до Лиссабона! Количество полётов: ' +
flightsVariantSecond);

  } else {

    console.log('Быстрей накопишь полётами в бизнесе до Валенсии! Количество полётов: ' +
flightsVariantFirst);

  }

}
```