

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение

высшего образования

**«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

ФАКУЛЬТЕТ СРЕДНЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ

Колледж экономики и информатики им. А.Н. Афанасьева

Выполненное задание на самостоятельную работу

по производственной практике «JavaScript»

Выполнил студент

Группы: ИСдо-31

Проконов Кирилл Сергеевич

Ульяновск

2023

**Основы JS**

* **Задание 1**

**Задание**: Нужно написать функцию, которая возвращает тип входного аргумента

**На входе**: один аргумент, любого значения

**На выходе**: строка, с типом входного аргумента

function getType(arg\_type) {

    return typeof arg\_type;

}

console.log(getType(undefined));  // "undefined"

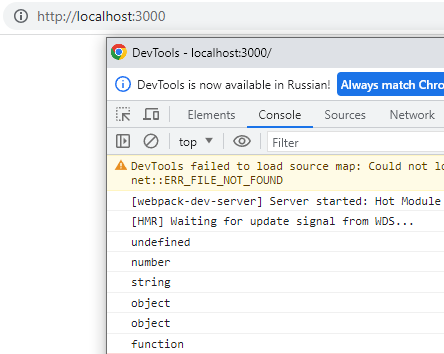
console.log(getType(5));  // "number"

console.log(getType("Kirill"));  // "string"

console.log(getType(null));  // "object"

console.log(getType([5, 8, 9]));  // "object"

console.log(getType(alert));  // "function"



* **Задание 2**

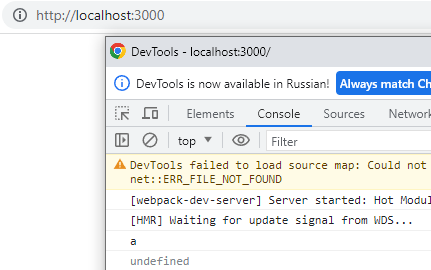
**Задание**: Что выведется в консоль и почему?

const a = () => console.log("a");

const b = () => console.log("b");

const c = a || b ? a() : b();

console.log(c);



В консоль выводится значение функции a и undefined.

Значение «a» прописалось, так как в константе «c» произошёл вызов функции “a()”. Функция вызвалась с помощью сравнения логическим оператором «или» и условным тернарнарным оператором. Логический оператор сделал проверку на истинное значение, и тернарный оператор вывел, то что ему было задано выводить при истине, в нашем случае это функция a.

Значение «undefined» выводится, так как выражение тернарного оператора a() : b()оценивается как результат выполненного вызова функции, который равен undefined. Функция a не имеет оператора возврата, поэтому она возвращает значение undefined по умолчанию.

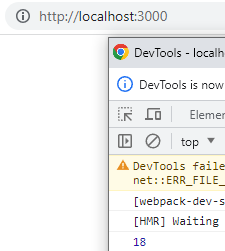
* **Функции**
* **Задание 1**

**Задание**: Нужно реализовать функцию, возвращающая сумму всех аргументов,

если тип аргумента - число. Вызовов ВСЕГДА 5

Выполнил два подхода. В первом, проверку ровно пяти аргументов. Во втором, проверку аргументов через цикл, аргументов может быть до бесконечности.

|  |  |
| --- | --- |
| function summa(a, b, c, d, e) {      let summa = 0;      if (typeof a === 'number') {          summa += a;      }      if (typeof b === 'number') {          summam += b;      }      if (typeof c === 'number') {          summa += c;      }      if (typeof d === 'number') {          summa += d;      }      if (typeof e === 'number') {          summa += e;      }      return summa;  }  const result = summa(5, true, 5, 8, null);  console.log(result); | function summa(arguments) {      let sum = 0;      for (let i = 0; i < arguments.length; i++) {          if (typeof args[i] === 'number') {              sum += args[i];            }      }      return sum;  }  const summa\_end = summa([5, true, 5, 8, null]);  console.log(summa\_end); |



* **Объекты и Массивы**
* **Задание 1**

**Задание**: Реализовать функцию *convertArrToObj*, которая на вход принимает

массив значений, а на выход отдает объект, где ключ - это индекс элемента в

массиве, а значение - сам элемент

function convertArrToObj(arr) {

    const obj = {};

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

      obj[i] = arr[i];

    }

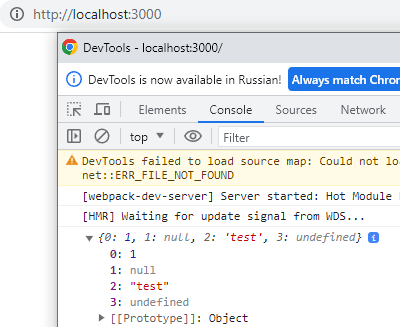
    return obj;

}

const arr = [1, null, 'test', undefined]

const obj = convertArrToObj(arr);

console.log(obj);



* **Задание 2**

**Задание**: Требуется реализовать функцию *countFromArr*, которая позволяет

подсчитать значения массива. На вход поступает массив, на выход получаем

объект, где ключ - это элемент массива, а значение - его кол-во в массиве

function countFromArr(arr) {

    const countObj = {};

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

      const object = arr[i];

      if (countObj[object]) {

        countObj[object] += 1;

      }

      else {

        countObj[object] = 1;

      }

    }

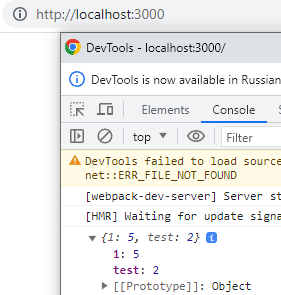
    return countObj;

}

const arr = [1, 1, 1, 1, 1, 'test', 'test'];

const res = countFromArr(arr);

console.log(res);



* **Задание 3**

**Задание**: Требуется реализовать функцию *groupByField*, которая позволяет

сгруппировать массив объектов по определенному полю этого объекта.

На вход поступает массив объектов и строка с названием нужного поля, на выход

объект - где ключом является значение поля объекта, а значением - массив

объектов где значение в поле равно этому ключу

function groupByField(arr, field) {

    const groupedObj = {};

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

      const object = arr[i];

      const obj\_value = object[field];

      if (!groupedObj[obj\_value]) {

        groupedObj[obj\_value] = [];

      }

      groupedObj[obj\_value].push(object);

    }

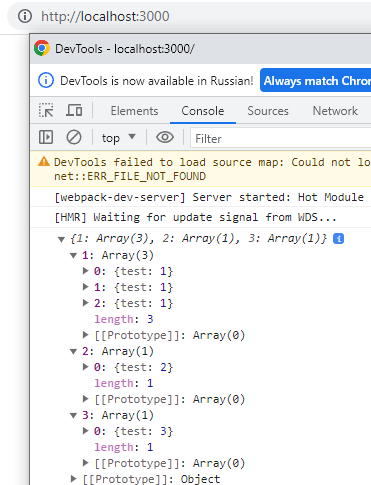
    return groupedObj;

}

const arr = [{test: 1}, {test: 2}, {test: 3}, {test: 1}, {test: 1}];

const res = groupByField(arr, 'test');

console.log(res);



* **Прототипы, наследование**
* **Задание 1**

**Задание**: Реализовать функцию plus*()*

*(1).plus(2); // 3*

function plus(number1) {

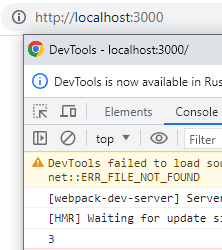
    return function(number2) {

      return number1 + number2;

    }

  }

console.log(plus(1)(2));



* **Асинхронная работа в JS"**
* **Задание 1**

**Задание**: Что выведется в консоль?

console.log("start");

setTimeout(() => console.log("timeout"), 0);

new Promise((resolve, reject) => {

console.log("promise constructor");

reject();

})

.then(() => console.log("p1"))

.catch(() => console.log("p2"))

.catch(() => console.log("p3"))

.then(() => console.log("p4"))

.then(() => console.log("p5"));

console.log("final");

Вначале выводится строка «start», после создания промиса выводится строка «promise constructor». После этого выводится строка "final". Внутри промиса также вызывается reject(), что вызывает ошибку, и поэтому после идёт её обработчик «catch» - p2. Остальные обработчики catch() пропускаются, так как уже был обработан отклоненный Promise в предыдущем catch() блоке. Выполняются блоки then(), в которых выводятся строки "p4" и "p5".

В самом конце выводится «timeout», так как у данного колбэка задана нулевая задержка, он будет выполнен после завершения текущего кода.

