

Функции в JavaScript

JS
COURSE
ORT DNIPRO

ORT**DNIPRO**.ORG/**JS**

1. Функции и функциональные выражения

Функции в JavaScript

```
2
3  function action(a, b, c){
4      let sum = a + b + c;
5      return sum;
6  }
7
8  let process = function(a, b, c){
9      let sum = a + b + c;
10     return sum;
11 }
12
13 let calculate = (a, b, c) => a + b + c;
14
15 typeof action; //function
16 typeof process; //function
17 typeof calculate; //function
18
```

Функции в JavaScript – блоки кода которые возможно вызывать (выполнять) многократно. Синтаксисом JS предусмотрено несколько способов определения функций: Объявление функции (***Function Declaration***) (3), Функциональное выражение (***Function Expression***, она же «анонимная» функция) (8), и стрелочные-функции (***arrow-function***, они же лямбда-функции) (13). Функции в JavaScript – тип данных, функцию мы можем размещать в переменных, как и другие типы данных. Отличие в том, что функции мы можем вызывать.

Оператор ... и функции

```
2
3   let process = function(a, b, c, ...others){
4       console.log(others);
5       let sum = a + b + c;
6       return sum;
7   }
8
9   process(1,2,3,4,5,6,7); // return 6;
10  // in console: [4,5,6,7];
11
```

Функция может принимать параметры и возвращать результат своей работы для дальнейшего использования (оператор *return*).

Но при помощи оператора ... (в данном случае его называют *rest-оператором*) мы можем принять любое количество параметров и работать с ними как с массивом (**ES2015**).

2. Таймеры в JavaScript

Таймеры в JavaScript

```
2
3   let f1 = function(){
4       |   console.log("Function for Timeout called");
5       |   }
6
7   let f2 = function(){
8       |   console.log("Function for Interval called");
9       |   }
10
11   let timeout_id = setTimeout(f1, 1000);
12
13   let interval_id = setInterval(f2, 3000);
14
```

setTimeout(*some_function*, *delay*) – вызовет функцию *some_function* через *delay* миллисекунд. Сделает это один раз.

setInterval(*some_function*, *delay*) – вызовет функцию *some_function* через *delay* миллисекунд. И будет повторять вызов каждые *delay* миллисекунд.

Обе функции возвращают **id** таймера, с помощью которого и функций **clearTimeout(id)** и **clearInterval(id)** уничтожить таймер еще до его вызова. Обе функции можно отнести к инструментам **асинхронности**.

3. Геолокация и **callback**'и

Геолокация в теории



Широта == Latitude

Долгота == Longitude

```
{ ..., latitude: 48.4767, longitude: 35.0543, ... };
```


Геолокация на практике

```
2
3 // 'Classic' version
4 navigator.geolocation.getCurrentPosition( position => {
5     console.log('Your position: ', position.coords);
6 }, error => {
7     console.log('Geolocation error:', error);
8 })
9
```

У браузера есть возможность узнать координаты пользователя на местности. Для этого мы можем воспользоваться методом **navigator.geolocation.getCurrentPosition()** который принимает **callback** функции для получения координат и информации об ошибке. Но важно **проверять поддерживает ли браузер геолокацию** проверяя наличие свойства **geolocation** объекта **navigator**.

Подробнее: <https://developer.mozilla.org/ru/docs/Web/API/Geolocation/getCurrentPosition>

Немного о статических карта на примере Here Map

```
https://image.maps.api.here.com/mia/1.6/mapview?app_id=oZmMWRV4tAjQmgkxBvF0&app_code=x5pKHqifhw1mnS_zBTIFsA&z=11&w=600&h=600&c=48.4608,35.0501
```

Сервис **Here Map** предоставляет возможность размещать на наших страницах картографические материалы, управляя позицией и масштабом отображения.

Вы можете воспользоваться шаблоном в репозитории [./src/template-geolocation/](#)

4. Перебирающие методы массивов

Метод .sort() и функция-компаратор

```
2
3   let arr = [23, 4, 67, 117, 34, 0, 55, 78, 5, 9];
4
5   arr.sort(function(a, b){
6       if(a > b){
7           return 1;
8       }else if(a < b){
9           return -1;
10      }else{
11          return 0;
12      }
13  });
14  //arr.sort((a,b) => a - b);
15
16  console.log(arr);
17  //[0, 4, 5, 9, 23, 34, 55, 67, 78, 117]
18
```

Методу **.sort()** массивов можно передать функцию (т.н. функцию-компаратор) которая «подскажет» браузеру как сравнивать два элемента между собой. Функция принимает 2 элемента и должна вернуть 0 если они равны, отрицательное число если второй элемент больше или положительное если первый элемент больше.

Полезные методы преобразования массивов

.filter();

Метод **.filter()** формирует новый массив занося в него элементы из старого, но только те которые «одобрит» функция переданная методу в качестве параметра.

.map();

Метод **.map()** формирует новый массив занося в него элементы из старого, но предварительно пропуская каждый элемент через функцию переданную методу в качестве параметра. Эта функция может любым образом преобразовать элемент.

Будет полезным

Перебирающие методы

В JavaScript есть еще ряд методов массивов, а именно: `.every()`, `.some()`, `.find()`, `.findIndex()`, `.forEach()`, `.reduce()` узнайте чем они могут быть полезны.

На следующем занятии

На следующем занятии



Домашнее задание
/сделать

Домашнее задание #С.1

Разработать скрипт, проверяющий знания (умение) **таблицы умножения двузначных чисел**. Скрипт должен задать пользователю 12 задач на умножение **двузначных** чисел. По результатам проверки, пользователю выставляется оценка (по 12 бальной шкале), а также выводиться два списка: верных ответов, и ошибочных ответов (*с указанием какой ответ был правильный*).

Домашнее задание #C2

Необходимо реализовать расчёт схемы выплаты кредита по **АННУИТЕТНОЙ** схеме. Параметры которые определяют кредит (данные на входе): **сумма**, **процентная ставка** (% годовых) и **срок кредитования** (в месяцах). Никакие комиссии и другие платежи не учитываем.

Месяц	Задолженность по кредиту	Погашение кредита	Проценты по кредиту	Комиссии	Выплаты в месяц
1	1000.00	83.40	20.90	0.00	104.30
2	916.60	83.40	19.10	0.00	102.50
3	833.20	83.40	17.40	0.00	100.80
4	749.80	83.40	15.70	0.00	99.10
5	666.40	83.40	13.90	0.00	97.30
6	583.00	83.40	12.20	0.00	95.60
7	499.60	83.40	10.50	0.00	93.90
8	416.20	83.40	8.70	0.00	92.10
9	332.80	83.40	7.00	0.00	90.40
10	249.40	83.40	5.20	0.00	88.60
11	166.00	83.40	3.50	0.00	86.90
12	82.60	82.60	1.80	0.00	84.40
Итого		1000.00	135.90	0.00	1135.90

В схеме должна быть информация за каждый месяц: сколько нужно погасить **тела кредита**, сколько заплатить **процентов** (и **сумму** тело + проценты) и сколько остаётся **долга** по телу кредите после этого платежа.

В конце необходимо вывести какая будет **переплата** по кредиту. Все денежные суммы должны быть округлены до 2х знаков после запятой.

Пример функционала:

<https://fin-calc.org.ua/ru/credit/calculate/>

Подробнее:

<https://finance.ua/ua/credits/sxemi-pogasheniya-kreditov>

<http://groshi-v-kredit.org.ua/yak-rozrahuvaty-rozmir-anujitetnoho-platezhu.html>