**Лабораторная работа №2**

**Функции и встроенные объекты JavaScript.**

Часто нам надо повторять одно и то же действие во многих частях программы.Чтобы не повторять один и тот же код во многих местах, придуманы функции. Функции являются основными «строительными блоками» программы.

В общем виде оформление функции должно соответствовать следующему формату:

**function** ИмяФункции (список формальных аргументов через запятые)

{

...

тело функции

...

**return** значение;

}

Оператор **return**, возвращающий значение функции, может быть не один, а может и вовсе отсутствовать. В случае если функция не возвращает никакого значения, ее вызов нельзя использовать в операторе присваивания и выражениях.

Вызов функции в общем виде выглядит так

**Имя Функции** (список фактических аргументов через запятую)

В качестве фактического аргумента функции может быть константа, переменная, выражение и, в частности, вызов другой функции.

JavaScript это объектно-ориентированный язык. Основной единицей в объектно-ориентированном языке является объект, который объединяет в себе данные (свойства) и средства обработки этих данных (методы). При интерпретации страницы HTML браузером создаются объекты JavaScript, свойства которых представляют значения атрибутов тэгов языка HTML. Одним из важных объектов является объект document. Объект **document** соответствует всему HTML-документу. Для вывода текста (в т.ч. фрагмента html) в один из элементов документа можно использовать свойство innerHTML.

**Пример:**

**HTML**

<span class="el">

</span>

**Javascript**

const el=document.querySelector(".el")//получаем ссылку на элемент с классом "el"

el.innerHTML="New text"

Можно выводить и значения переменных и выражений, используя обратные кавычки

**Пример:**

const el=document.querySelector(".el")

const a=5

el.innerHTML=`value= ${a}`

**Функция как тип данных**

Служебное слово **function** определяет переменную с именем f, т.е функция – это тип данных.

**function** f(arg1,arg2)

{

/\* телофункции \*/

}

Console.log('Тип переменной f: '+ typeof(f));

После выполнения функции на консоль выведется: Тип переменной f: function

**Функция как объект**

В результате выполнения фрагмента представленного будет выведено сначала значение переменной i: 5, а затем будет выведено значение переменной f

<script language="JavaScript">

var i=5;

**function** f(a,b,c)

{

if (a>b) return c;

}

Console.log('Значение переменной i: '+ i.valueOf());

Console.log('<BR>Значение переменной f:<BR>'+ f.valueOf());

**Встроенные функцииJavaScript**

Рассмотрим встроенные функции JavaScript, которые могут пригодиться при написании программ.

**parseInt**(строка) - преобразует указанную строку в целое число

**Пример:**

a =parseInt("2.5")

b =parseInt("10 apples")

**parseFloat**(строка) - преобразует указанную строку в число с плавающей разделительной (десятичной) точкой

b =parseFloat("10 apples")

d =parseFloat("-15.87")

**isNaN** (значение) - возвращает true, если указанное в параметре значение не является числом, иначе - false

a =isNaN("2.5")

b =isNaN("10 apples")

d =isNaN("-15.87")

**eval** (строка) - вычисляет выражение в указанной строке.

a=5

eval("if (a<10) { a + 2 }")

**//** переменная а примет значение a=7

**Встроенные объекты**

Встроенные объекты имеют фиксированные названия, свойства и методы. Свойства имеют имена и значения. Значение большинства свойств можно менять.

### *Объект String (строка)*

Для создания строкового объекта используют выражение следующего вида

имя\_переменной = newString ("строковое\_значение")

Создать строковый объект и с помощью обычного оператора присваивания

имя\_переменной = "строковое\_значение"

Одним из важных свойств строкового объекта является свойство length позволяющее определить длину строки

str = "Привет мир!"

x = str.length

console.log(x)

**МетодыобъектаString:**

|  |  |
| --- | --- |
| **Метод** | **Описание** |
| charAt(ind) | Возвращает символ из указанной позиции строки (нумерация с нуля) |
| [toLowerCase()](https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Global_Objects/String/toLowerCase) | Переход к нижнему регистру (к строчным символам) |
| [toUpperCase()](https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Global_Objects/String/toUpperCase) | Переход к Верхнему регистру (к заглавным символам) |
| indexOf(subStr,startInd)  indexOf(subStr) | Поиск подстроки subStr, начиная с позиции startInd и до конца строки. Метод возвращает позицию, с которой начинается подстрока. Если startInd не задан, поиск начинается с нулевой позиции. Если подстрока не найдена, метод возвращает -1 |
| [includes(substr, pos](https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Global_Objects/String/includes)) | возвращает true, если в строке str есть подстрока substr, либо false, если нет |
| [startsWith](https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Global_Objects/String/startsWith)([substr](https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Global_Objects/String/includes)) | проверяет начинается ли строка со строки [substr](https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Global_Objects/String/includes) |
| [endsWith](https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Global_Objects/String/endsWith)([substr](https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Global_Objects/String/includes)) | проверяет заканчивается ли строка строкой [substr](https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Global_Objects/String/includes) |
| slice(start [, end]) | Возвращает часть строки от start до (не включая) end. Если аргумент end отсутствует, slice возвращает символы до конца строки |
| substring(start [, end]) | Возвращает часть строки между start и end |
| substr(start [, length]) | Возвращает часть строки от start длины length. |
| trim() | убирает пробелы в начале и конце строки |
| строка1.concat(строка2) | Склеивание строк |

# ***Объект Array (массив)***

Существует несколько способов создания массива. В любом случае, прежде всего, создается новый объект массива:

имя\_массива = new Array ([длина массива])

Массив также можно создать по умолчанию, например, при таком описании:

**Var** set = ["сложение","вычитание","умножение","деление"];

Объект массив также как и строка обладает свойством length. Длину массива можно определить, например, следующим образом

var set= new Array(0,1,2,3,4,5,6,7,8,9,10)

alert(set.length);

|  |  |
| --- | --- |
| **Метод** | **Описание** |
| pop | удаляет последний элемент из массива и возвращает его |
| push(...items) | добавляет элемент в конец массива |
| shift() | извлекает элемент из начала массива |
| unshift(...items) | добавляет элементы в начало массива |
| indexOf(item, from) | ищет item, начиная с индекса from, и возвращает индекс, на котором был найден искомый элемент, в противном случае -1 |
| lastIndexOf(item, from) | то же самое что indexOf, но ищет справа налево |
| includes(item, from) | ищет item, начиная с индекса from, и возвращает true, если поиск успешен |
| [reverse](https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Global_Objects/Array/reverse)() | меняет порядок элементов массив на обратный. |
| [split(delim)](https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Global_Objects/String/split) | разбивает строку на массив по заданному разделителю delim |
| [join(glue](https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Global_Objects/Array/join)) | делает в точности противоположное split, т.е. создаёт строку из элементов массива, вставляя разделитель glue между элементами |
| concat(array) | возвращает массив, полученный добавлением массива array. Исходный массив не меняется |
| sort(function)  sort() | сортировка массива. Параметром является имя функции, которая задает правила сравнения двух элементов. Если параметр опущен, элементы сортируются в лексикографическом порядке |
| filter(function(item, index, array) {}); | возвращает новый массив с элементами, которые соответствуют правилам фильтрации |
| map(function(item, index, array) {}) | создаёт новый массив с результатами вызова указанной функции на каждом элементе данного массива |
| forEach(function(item, index, array){}); | применяет передаваемую в качестве его аргумента функцию к массиву. Функция, передаваемая методу forEach, сама принимает два аргуменста: элемент на текущем шаге итерации и текущий индекс |

***Объект Math*:**

Объект Math предназначен для хранения некоторых математических констант (например, число пи) и выполнения преобразования чисел, и вычисление значений типичных математических функций. Работа со свойствами и методами объекта Math происходит следующим образом:

|  |  |
| --- | --- |
| **метод** | **описание** |
| abs(arg) | Возвращает абсолютное значение аргумента. |
| sin(arg)  cos(arg)  tan(arg) | Стандартные тригонометрические функции, аргумент указывается в радианах. |
| asin(arg)  acos(arg)  atan(arg) | Обратные тригонометрические функции, возвращают значения в радианах. |
| exp(arg)  log(arg) | Экспонента и натуральный логарифм (основание e). |
| ceil(arg) | Возвращает ближайшее целое число, большее или равное аргументу. |
| floor(arg) | Возвращает ближайшее целое число, меньшее или равное аргументу. |
| min(arg1,arg2)  max(arg1,arg2) | Минимальное (максимальное) значение из двух числовых аргументов. |
| pow(arg1,arg2) | Возвращает arg1 в степени arg2. |
| random() | Возвращает случайное число из диапазона [0,1]. |
| round(arg) | Округляет аргумент до ближайшего целого. |
| sqrt(arg) | Корень квадратный из аргумента. |

***Объект Date(Дата)***

Создать объект с текущей датой и временем можно таким образом:

**var a = newDate();**

Также можно создать объект с заданной датой и точным времени. Формат такого конструктора выглядит так:

**newDate(год,месяц, день [,часы [,минуты [,секунды [,мс]?]?]?]?)**

Параметры, указанные в квадратных скобках не являются обязательными

**Параметры:**

год — числовое выражение, задающее полный номер года (например, 1988, а не 88);

месяц — числовое выражение, задающее номер месяца (0 = январь, …, 11 = декабрь);

день — числовое выражение, задающее номер дня в месяце от 1 до 31;

часы — необязательное числовое выражение, задающее номер часа от 0 до 23;

минуты — необязательное числовое выражение, задающее номер минуты от 0 до 59;

секунды — необязательное числовое выражение, задающее номер секунды от 0 до 59;

мс — необязательное числовое выражение, задающее номер миллисекунды от 0 до 999.

Для примера создадим объект с датой 18 февраля 2014 г.18 часов10 минут

New Date(2014, 1, 18, 18, 10)

**Методы объекта Date**

|  |  |
| --- | --- |
| **метод** | **описание** |
| getYear() | Возвращает год. Значение года, принадлежащее XX веку, возвращается в виде двух последних цифр. |
| setYear(year) | Устанавливает год. |
| getMonth() | Возвращает месяц. |
| setMonth(month) | Устанавливает месяц. |
| getDate() | Возвращает день месяца. |
| setDate(day) | Устанавливает день месяца. |
| getDay() | Возвращает день недели (0-воскресенье, 1-понедельник, ... 6-суббота). |
| getHours() | Возвращает час. |
| setHours(hours) | Устанавливает час. |
| getMinutes() | Возвращает минуты. |
| setMinutes(minutes) | Устанавливает минуты. |
| getTime() | Возвращает число миллисекунд прошедших с 0 часов 1 января 1970. |
| setTime(time) | Устанавливает число миллисекунд прошедших с 0 часов 1 января 1970. |

**Задания:**

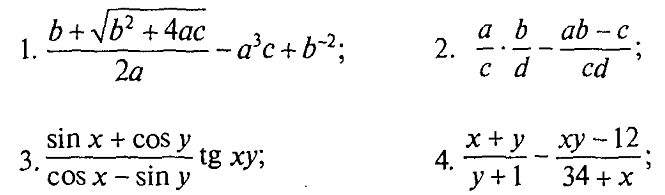
1. Разработать функцию, определяющую наименьшее число из 3-х введенных чисел *a,b,c (метод* ***Math.min*** *не использовать!)*. Функция должна выводить значения на HTML страницу в следующем виде, например:

**Введены значения:**

* + 1. a=2
    2. b=3
    3. c=4

**Наименьшее число:** a=2

1. Треугольник задан координатами своих вершин. Разработать функцию вычисляющую длину стороны треугольника (*формула длины отрезка по координатам*). Также разработать функции для вычисления площади и периметра треугольника, использующие функцию вычисления стороны треугольника. Вывести результат вычислений на HTML страницу.
2. Разработать функции, которые будут вычислять значение выражения, по заданным через параметры переменным

****

1. Пользователь вводит через пробел двадцать слов, через диалоговое окно prompt. Вывести на страницу список *в исходном виде*, *в обратном порядке* и *отсортированный список*. Слова, начинающиеся с гласной буквы выделить жирным шрифтом. Также вывести список слов начинающихся с согласной буквы (для получения списка таких слов использовать метод массива filter)
2. Анкета. Сначала создается массив вопросов не менее 10 вопросов. Далее с помощью **диалоговых окон**!!! идет опрос анкетируемого. Вычисляется возраст анкетируемого, дата и время анкетирования. После опроса все анкетные данные выводятся на страницу. Анкетные данные оформляются в виде таблицы. (Ввод имени фамилии и отчества производится в именительном падеже, вывод в документ должен начинаться с со Строки «Анкета ФИО в род. падеже»).