Основы JavaScript

Алексей Тарасов

http://testhost.ru/
https://developer.mozilla.org/ru/docs/Web/JavaScript - Документация
http://htmllab.ru/zadachi-po-javascript/ - Домашние задания (проверка в классе)
https://vk.com/jsspec - группа поддержки
https://www.youtube.com/htmllabru - канал

Введение в JavaScript

- интерпретируемый объектно-ориентированный
- ECMAScript
- Unicode

Применение языка

- браузеры
- сервер
- обработка форм Adobe Reader
- автоматизация Photoshop
- смартфоны

Обзор базовых типов

- string "привет" 'привет' `привет`
- number 12 34.6 1e5
- boolean true false
- null, undefined
- object { props:"some" }
- array [23, 45, 100]
- function ...

Операторы

- арифметические операторы +, -, /, *, %, **
- оператор группировки ()
- строковый +

- определение типа typeof
- сравнения < > <= >= != == ===
- логические операторы ! && ||
- операторы присвоения = += -= *= /= %= ++ --

Приоритеты операторов

1. ()
2. постфиксные ++ --
3. логическое отрицание !, унарные операторы, префиксные ++ --, typeof
4. **
5. * / %
6. + -
7. побитовые сдвиги
8. < <= > >= in
9. == != === !==
10. &&
11. ||
12. += *= /= %= -=

Примеры применения операторов

2 + 4
"Привет, " + 'мир'
10 < 2
true && false
34 % 2
2 ** 3 (то же самое, что Math.pow(2,3))

Памятка по логическим операторам

&& логического умножения И
false && false -> false
true && false -> false
false && true -> false
true && true -> true

|| логическое сложение ИЛИ
false || false -> false
true || false -> true
false || true -> true
true || true -> true

Практическая работа

```
(5 && ("с п е ц и а л и с т" || !0)) && !(0 && (56 || true))
```

Выражения и инструкции

- 45 + "кг"
- 45 + "кг";

Переменные и константы

- var let const
- алфавитно-цифровые символы $ _

Манипуляции с базовыми типами

- var a = "5";
  a = a * 1 или a = +a или parseInt(a) или parseFloat(a)
- var d = 5;
  d = d + "" или d = new String(d) или d = d.toString()
- var z = true;
  z = z + "" или z = new String(z)

"Маша" < "Медведь"

Тривиальные типы

- null, undefined
- typeof t == "string"

Подключение JS-скрипта

```
<script> /* т у т  JavaScript- к о д */ </script>
<script src="js/hello_world.js"></script>
```

Практические работы

1. Задание

```
/*Н а й т и  п л о щ а д ь  п р я м о у г о л ь н и к а  ш и р и н о й  5  и
в ы с о т о й  10*/
```

```
var
  S = 0, //площадь
      width = 5, //ширина
      height= 10;//высота

S = width * height;
console.log("Площадь: " + S);
console.log(`Площадь: ${S}`);
```

2. /*Найти периметр прямоугольника шириной 5 и высотой 10*/

```
let
  width = 5,    //ширина
  height = 10,  //высота
  perimeter = 0 //периметр
;

perimeter = 2 * (width + height);
console.log("Периметр: " + perimeter);
```

3. /*Найти периметр квадрата с площадью 25*/

4. /*Найти диагональ квадрата с площадью 25*/

5. Задание

```
/*Найти индекс массы тела I, при массе
человека 100кг, а его рост 1.8м */
let
  I = 0,  //индекс массы тела
  m = 100,//масса тела в кг
  h = 1.8;//рост в метрах

I = m / (h * h);
console.log(`I = ${I}`);
```

Задача: верблюд двигается со скоростью 10км/ч. Через какое время они пройдет 5.4 км? 32.4

Задание на квадратное уравнение

```
/*Дано квадратное уравнение
4x*x-14x+10 = 0. Найти значение x
```

```
*/
let
 a = 4,  //старший коэффициент
 b = -14,  //средний коэффициент
 c = 10, //свободная член
 D = 0,  //дискриминант,
 x1, x2  //переменные для рез-та

D = b * b - 4  * a * c;//так находится дискриминант
console.log(D); //выведем его
console.log(D > 0); //точно больше нуля? true

x1 = (-b + Math.pow(D,1/2))/(2*a);
x2 = (-b - Math.pow(D,1/2))/(2*a);

console.log(`x1=${x1},x2=${x2}`); //решение
```

Задание на квадратное уравнение

Тест: типы данных, операторы, переменные

https://goo.gl/forms/EcGkok25wE6KJ4E42

Управляющие конструкции if – else if – else

```
if ( ВЫРАЖЕНИЕ )
  ОПЕРАТОР;

if ( ВЫРАЖЕНИЕ ){
  ОПЕРАТОР1;
…
  ОПЕРАТОРN;
}

if ( ВЫРАЖЕНИЕ ){
  …
} else {
  ...
}
```

Пример IF

```
var s = 125000;

if ( s < 115000){
      console.log("У ч и т ь с я");
} else {
      console.log("Р а б о т а т ь");
}

if ( typeof s == "number" ) {
      console.log("о к , р а б о т а е м  с  ч и с л о м");
} else {
      console.log("н е  о к , н е т  ч и с л а");
}
```

Тернарный оператор

ВЫРАЖЕНИЕ ? ЕСЛИtrue : ЕСЛИfalse

```
//т е р н а р н ы й  о п е р а т о р
console.log(s < 115000 ? "У ч и т ь с я" : "Р а б о т а т ь");
```

Практическая работа

1. Задание

```
/* Создать переменную s со значением "test"
 Проверить тип переменной
 Если переменная строковая, напечатать
"Ошибка ввода"
 Если переменная числовая, напечатать
квадрат числа
*/
var s = "test";

if( typeof s == "number" ) {
  console.log(s*s);
} else {
  console.log("Ошибка ввода");
}
```

2. Задание

```
/*Определить является ли число 3243 чётным
или нечётным. Сообщить о результате через
консоль*/


const num = 3243;
var   message = "";
if( num % 2 ){
  message = "нечётное";
} else {
  message = "чётное";
}
//message = num % 2 ? "нечётное" : "чётное";
console.log(`${num} - ${message}`);
```

3. Задание

```
/*Определить является ли человек
совершеннолетним. В переменную age
поместить любое число (положительное, не
более 100)*/
```

4. Задание

```
/*Даны два числа a и b. Напечатать то, которое меньше. Если они равны, напечатать "числа равные"*/


const
  a = Math.round(Math.random()*10),
  b = Math.round(Math.random()*10);
let result = "";

if( a < b ){
 result = a;
} else if( a == b ){
 result = "числа равны";
} else {
 result = b;
}
//result = a < b ? a : a == b ? "равны" : b;
console.log(`a=${a},b=${b},result=${result}`);
```

5. Задание

```
/* Написать правильно фразу "В корзине N товаров"
 товар   1, 21, 31...
 товара  2-4, 22-24...
 товаров 0, 5-20, 25-30...
*/
var goods = 15, word = "товаров";
if ( goods % 100 < 5 || goods % 100 > 20 ) {

  if( goods % 10 == 1 ) word = "товар";
  if( goods % 10 > 1 && goods % 10 < 5)
          word = "товара";
 }
console.log(`В корзине ${goods} ${word}`);
```

switch

switch (ВЫРАЖЕНИЕ) {
  case ЗНАЧЕНИЕ1: ОПЕРАТОР1; break;
 …
  case ЗНАЧЕНИЕN: ОПЕРАТОРN; break;

```
  default: ОПЕРАТОРN+1;
}
```

Пример switch

```
var day = 1, menu = "";

switch( day ){
      case 1: menu = "г р е ч к а"; break;
      case 2: menu = "с е ч к а"; break;
      case 3: menu = "о в с я н к а"; break;
      case 4: menu = "я ч к а"; break;
      case 5: menu = "г е р к у л е с"; break;
      case 6: menu = "п ш е н к а"; break;
      default: menu = "о л и в ь е";
}

console.log(menu);
```

Практическая работа

1. Задание

```
let monthIndex = 11, monthName = "";
monthIndex = Math.round(Math.random()*11);

switch( monthIndex % 12 ) {
      case 0: monthName = "я н в а р я"; break;
      case 1: monthName = "ф е в р а л ь"; break;
      case 2: monthName = "м а р т"; break;
      case 11: monthName= "д е к а б р ь"; break;
}

console.log(`и н д е к с  = ${monthIndex}, monthName=${monthName}`);
```

2. Задание

```
/* * В ы с о т а h р а в н а 1.5, а  д л и н а  о с н о в а н и я  ( е с л и
е с т ь ) d - 3.4. Н а й т и  п л о щ а д ь  ф и г у р ы  п о  е ё  т и п у
type (square, circle, rectangle, triangle) */


3
```

Задание 3
Напишите скрипт, который найдет стоимость товара по исходной стоимости и по дневному коэффициенту. понедельник - 1.1, вт - 1.7, ср - 1.3, чт - 2, пт - 1.5, 1

Операторы инкремента и декремента

```
var i = 0;
//i++;
console.log(i++);

var i = 0;
//++i;
console.log(++i);
```

Цикл while

```
while ( ВЫРАЖЕНИЕ ){
  ОПЕРАТОР1;
}
```

Пример while

```
var i = 0;
while( i < 10 ){
  console.log(i);
  i++;
}
```

Практическая работа

1. Задание 1 /* Используя while выведите все числа от 1 до 20 */
2. Задание 2 /* Выведите все нечётные числа от 1 до 20 */
3. Задание 3 /* Выведите квадраты всех чисел от 1 до 20 */
4. Задание 4 /* Выведите результаты умножения числа 8 на числа от 1 до 10 */
5. /*Автомобиль движется с постоянной скоростью v = 90. Выводить пройденный путь каждый интервал времени t, пока не проедем расстояние в 1000 */
6. /*Найти наибольшую степень двойки до 1000*/
7. * Задание 7

```
/* С ф о р м и р у й т е  и  в ы в е д и т е  т а б л и ц у*/
```

```
var table = "<table border=1>";
var i = 0, j = 0;
while( i < 10 ){
      table += "<tr>";
      i++;
      j=0;
      while( j < 10){
            table += "<td>";
            table += "<sup>"+i+"</sup>"+j;
        j++;
      }
}
table += "</table>";
//console.log(table);
```

8. * Задание 8

```
 /*Создайте "игру" */
var
  m = 55555, //наша попытка угадать число
  n = 5, //количество цифр в числе, которое
загадывает скрипт
  i = 0,  //просто счётчик
  ticket = ""; //число, которое загадывает скрипт

while( i < n ){
 ticket += Math.round(Math.random()*9);
 i++;
}
console.log(ticket);
if(ticket == m){
      console.log("Вы победили!");
} else {
      console.log("Попробуйте ещё!");
}
```

Цикл for

for (СОЗДАЕМ СЧЕТЧИК ; ПРОВЕРЯЕМ ; МЕНЯЕМ){
  ОПЕРАТОР;
}

Пример цикла for

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

Практическая работа

1. Задание

```
 /* Используя for выведите все числа от 1 до 20
*/
```

2. /* Выведите в консоли "ёлку":
   *
   ***
   *****
   ****

   */
3. * Задание

```
   /* Постройте таблицу умножения 10x10 в
консоли*/
```

4.

```
<ul>
 <li>Lorem.</li>
 <script>
 for(let i = 0; i < 4; i++)
  document.write( "<li>" + i)
 </script>
</ul>
```

Тест: конструкции языка

https://goo.gl/forms/aWRaouPp6ddbaEqJ3

День 2

Понятие функций

- механизм повторного использования кода
- специальный тип объектов формализующий логику поведения
- способы объявления
- способы вызова
- параметры и аргументы вызова
- область видимости
- возвращаемое значение

Способы объявления

- декларативный стиль

```
function ИМЯФУНКЦИИ(АРГУМЕНТ1[,...]){
  ТЕЛО ФУНКЦИИ;
  [return ЗНАЧЕНИЕ;]
}
ИМЯФУНКЦИИ(АРГУМЕНТ1[,...])
```

- функциональный стиль
- стрелочные функции

Аргументы по умолчанию

```
function ИМЯФУНКЦИИ(АРГУМЕНТ1 = ЗНАЧЕНИЕ1, ...){
  ТЕЛО ФУНКЦИИ;
  [return ЗНАЧЕНИЕ;]
}
```

Пример декларативного описания

```
function ipoteka(S,p,n){
  p = p / 1200;
  n = n * 12;
  return S * p / (1 - Math.pow(1+p,-n)) ;
}

var t = ipoteka(1e6,10,10);
console.log(t);
```

Практическая работа

1. Задание

```
/*Написать функцию square() для нахождения площади прямоугольника со сторонами width и height*/

function square(width, height){
	//console.log(width * height);
	return width * height;
}
var s = square(5,10);
console.log(s);
```

2. Задание

```
/*Напишите функцию sign(t), которая принимает аргумент t и возвращает 1, если число t - положительное, и -1, если t - отрицательное. В противном случае - 0 */
```

3. Задание

```
/*Написать функцию rand(), которая возвращает случайное целое число*/

function rand(n1,n2){
	return Math.round(Math.random()*(n2 - n1)+n1);
}
console.log(rand(10,20));
```

4. Задание

```
/*Написать функцию ipoteka, для подсчета аннуитетных ипотечных платежей. Функция принимает следующие параметры S - тело кредита, p - процентная ставка за год n -количество лет, на которые берется кредит*/

function ipoteka(S,p,n){
	p = p / 12;
	p = p / 100;
	n *= 12;
	return S * p / (1 - Math.pow(1+p,-n));
}
```

```
console.log(ipoteka(6e6,10,5));
```

5. Задание : написать функцию индекса массы тела (https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D0%B4%D0%B5%D0%BA%D1%81_%D0%BC%D0%B0%D1%81%D1%81%D1%8B_%D1%82%D0%B5%D0%BB%D0%B0)

```
/**
 * Находит индекс массы тела
 * @var number m масса тела
 * @var number h высота в метрах
 */
let bmi = (m, h) => m / (h * h);
console.log(bmi(120,1.6));//46.87

let bmi2 = (m, h) => {
    const tmp = m / (h * h);
    if(tmp <= 16) return "Выраженный дефицит массы тела";
    if(tmp <= 18.5) return " Недостаточная (дефицит) масса тела";
    if(tmp <= 24.99) return "Норма";
    if(tmp <= 30) return "Избыточная масса тела (предожирение)";
    if(tmp <= 35) return "Ожирение";
    if(tmp <= 40) return "Ожирение резкое";
    return "Очень резкое ожирение";

}
console.log(bmi2(120,1.6));//Очень резкое ожирение

var bmi3 = (m, h) => {
   const tmp = m / (h * h);
   if (tmp <= 18.5)   return "нужно набирать " + ( 18.5 * h * h - m);
   if (tmp > 24.99)  return "нужно сбрасывать " + (m - 24.99 * h * h);
   return "нормальный вес";
}
console.log(bmi3(120,1.6));
```

6. dd

Функциональный стиль описания

var foo = function([АРГУМЕНТЫ]){

ТЕЛО ФУНКЦИИ;
}

Пример функционального стиля описания

```
var ipoteka = function(S,p,n){
  p = p / 1200;
  n = n * 12;
  return S * p / (1 - Math.pow(1+p,-n)) ;
};
console.log(ipoteka(2e6,10,10));
```

Анонимная функция

- `(function(a,b){return a - b;})(10,3)`

Стрелочные функции

(АРГУМЕНТ1[,...]) => { ТЕЛО ФУНКЦИИ }
(АРГУМЕНТ1[,...]) => ВЫРАЖЕНИЕ

Примеры стрелочной функции

```
(a,b) => a * b

var ipoteka = (S,p,n) => {
  p = p / 1200;
  n = n * 12;
  return S * p / (1 - Math.pow(1+p,-n)) ;
};
console.log(ipoteka(2e6,10,10));

var ipoteka = (S,p,n) => S * (p/1200) / (1 - Math.pow(1+p/1200,-n*12));
console.log(ipoteka(2e6,10,10));
```

Область видимости

- глобальная и локальные
- аргументы и область видимости

Замыкания

```
console.clear();
var x = 10;

function foo(x){
  return function(){
    return x + 2;
  };
}

var t = foo(100);
console.log(t);
console.log(t());
```

Пример замыкания

```
function func1(){
  var var1 = 100;
  return function (){
    return var1++;
  }
}
let d = func1();
console.log( d() );
console.log( d() );
console.log( d() );
```

```
function game(){
       var i = Math.round(Math.random()*10);
       return function (n){
        if( i == n ){
          console.log("П о б е д а");
          i = Math.round(Math.random()*10);
        }else{
          console.log("П о п р о б у й т е  е щ ё ...");
        }
       }
}
```

```
let go = game();
```

arguments

- псевдопеременная доступа к аргументам
- arguments[0]
- arguments.length
- arguments.callee

Практическая работа

1. Задание

```
/* Написать функцию, которая находит
среднее арифметическое */

function avg(){
	var s = 0;
	console.log(arguments.length);
	console.log(arguments[2]);

	for(let i = 0; i < arguments.length; ++i)
		s += arguments[i];

	return s / arguments.length;
}
//		0 1  2  3   4
console.log(avg(3,4,10,34,100));
```

2. * Задание

```
/*Напишите игровой пример с замыканием.
Игрок должен вызывать функцию, замкнутую
на переменной */

function game(n){
	return function(m){
		if(n == m)
			console.log("Победа!");
		else
			console.log("Пробуйте..");
	}
```

```
}
var mygame = game(rand(0,10));

function game(n){
	return function(m){
		console.log(n == m ? "П о б е д а!" : "П р о б у й т е..");
	}
}
```

3. Задание /* Найти кол-во цифр в числе */
4. Задание /* Написать функцию нахождения площади прямоугольника. Например, может использоваться для нахождения площадь паркета */
5. * Задание /* Расчет количества ракетного топлива для доставки M количества груза на околоземную орбиту */
6. Задание /* Создать функцию, которая правильно выводит фразу "N товаров" */
7.

Рекурсия (если будет время)

```
function d(n){
	if(n > 0){
		d(n-1);
		console.log(n);
	}
}
d(3);
```

Тест по функциям

https://goo.gl/forms/OKdydGfjYmhDKt3F2

Объектный тип: Объект (Object)

```
var o = { props1: value1, [, ...] }
console.log(o.props1);
```

Свойства объекта

```
var o = {
 firstName:"В а с я",
 age:23
};
o.lastName = "О л е г о в";
o["prof"]  = "п р о г р а м м и с т";

console.log(o);
console.log(o["lastName"]);
console.log(o.prof);
```

Методы объекта

```
var o = {
 firstName:"В а с я",
 age:23,
 say: function (){
   console.log(o.firstName)
 }
};
o.say();
```

Копирование по ссылке

```
var a = 1, b;
b = a;
a = 5;
console.log(a,b);//5 1

var a = {n:1}, b;
b = a;
a.n = 5;
```

```
console.log(a.n,b.n);//5 5
```

in

- оператор проверки вхождения св-ва
- if("firstName" in o) console.log(o["firstName"]);

this

- ссылка на объект вызова

```
var o = {
 firstName:"Вася",
 age:23,
 say: function (){
   if( "firstName" in this )
     console.log(this["firstName"]);
 }
};
o.say();
```

Практическая работа

1. Задание

```
/* Создайте объект с двумя простыми свойствами и функциональным свойством (методом), выводящим эти свойства. Вызовите метод объекта. */
```

Пример this на разные объекты

```
console.clear();
var petya = {
 firstName:"Петя",
 age:32,
 boom: function(){
   console.log("Бьёт "+this.firstName);
 }
}
var olya = {
 firstName:"Ольга",
 age:32,
 boom: petya.boom
```

```
}
olya.boom();
```

for/in и проход по свойствам

```
for(let props in obj){
  console.log(props, obj[props]);
}
```

Пример for/in

```
var o = {
 firstName:"В а с я",
 age:23,
 say: function (){
   for(let p in this)
    //if(typeof this[p] != "function")
     console.log(p+" = "+this[p]);
 }
};
o.say();
```

toString()

```
 toString: function(){
   return "Ч Т О - Т О  С В О Ё";
 }
```

Практическая работа

1. Задание

```
/* С о з д а й т е  и  о п и ш и т е  у  о б ъ е к т а  м е т о д toString()
т а к ,  ч т о б ы  к о г д а  о б ъ е к т  в ы з ы в а л с я  в
ф р а г м е н т е  к о д а console.log(obj + ""), в  с к о б к а х
в ы в о д и л и с ь  е г о  с в о й с т в а .  П о д с к а з к а :
и с п о л ь з у й т е for/in */
```

Методы функций apply и call

```
function getProps(){
      for(let p in this){
        console.log(p + "=" +this[p]);
  }
}
getProps.apply(user);
```

Тест по объектам

https://goo.gl/forms/y78WxsA3Kc8j32AW2

Объектный тип: Массив (Array)

```
var arr = [ 2, "п р и в е т", true, function(){}, {}, [] ];
var arr = [ 8, 4, 5, 7, 1];

arr[ 5 ] = 6;
arr[ 15 ] = 10;
```

Свойство и методы массивов

```
.length
```

- **concat()** объединяет два массива и возвращает новый массив
- **join(deliminator = ',')** объединяет элементы массива в текстовую строку
- **push()** добавляет один или несколько элементов в конец массива и возвращает результирующую длину
- **pop()** удаляет из массива последний элемент и возвращает его
- **shift()** удаляет из массива первый элемент и возвращает его
- **unshift()** добавляет один или несколько элементов в начало массива и возвращает его новую длинну.
- **slice(start_index, upto_index)** возвращает секцию массива как новый массив
- **splice(index, count_to_remove, addElement1, addElement2, ...)** удаляет часть элементов из массива и (опционально) заменяет их. Возвращает удалённые элементы
- **replace()** - замена одной строки - другой строкой
- **reverse()** переставляет элементы массива в обратном порядке: первый элемент становится последним, а последний - первым
- **sort()** сортирует элементы массива
- **indexOf(searchElement[, fromIndex])** ищет в массиве элемент со значением searchElement и возвращает индекс первого совпадения
- **lastIndexOf(searchElement[, fromIndex])** тоже самое, что и indexOf, но поиск ведется в обратном порядке, с конца массива
- **forEach(callback[, thisObject])** выполняет callback-функцию по каждому элементу массива
- **map(callback[, thisObject])** возвращает новый массив, содержащий результаты вызова callback-функции для каждого элемента исходного массива
- **filter(callback[, thisObject])** возвращает новый массив, содержащий только те элементы исходного массива, для которых вызов callback-функции вернул true
- **every(callback[, thisObject])** возвращает true, если вызов callback-функции вернул true для всех элементов массива
- **some(callback[, thisObject])** возвращает true, если вызов callback-функции вернет true хотя бы для одного элемента

- **reduce(callback[, initialValue])** последовательно применяет callback-функцию callback(firstValue, secondValue) для того, чтобы свести все элементы массива к одному значению. В первый параметр функции передаётся предыдущий результат работы функции или первый элемент, а во второй - текущий элемент. Третьим параметром передаётся индекс текущего элемента
- **reduceRight(callback[, initalvalue])** работает так же как и reduce(), но порядок обхода ведется от конца к началу.

Практическая работа

1. Задание

```
/* Создайте массив из пяти чисел и выведите его элементы */
```

2. Задание

```
/* Создайте массив из сто пяти случайных чисел и выведите его элементы */
```

```
3. Задание Отсортируйте массив из 105 элементов
```

Пример использования метода

```
var arr = [ 3, 24, 11, 56, 7 ];
arr.sort((a,b) => a - b);
console.log(arr);

var arr = [ "Паша", "Яна", "Аня", "Нина" ];
arr.sort();
console.log(arr.reverse());
```

Сортировка массива объектов с замыканием

https://www.youtube.com/watch?v=8JsteOQxJZc

Тест на массивы

https://goo.gl/forms/Hfe3CGpspuW88wOK2

День 3

Конструкторы объектов

- функция для создания объектов, вызываемая с new

```
function City(name,population){
  this.name = name;
  this.population = population;
}
let c1 = new City("М о с к в а",1.1e7);
console.log(c1)
```

Прототипы

- это объект, свойство конструктора
- реализует наследование
- реализует цепочку поиска метода

```
function City(name,population){
  this.name = name;
  this.population = population;
}
City.prototype.show = function(){
  for(let i in this)
    console.log(`${i}=${this[i]}`);
}
let c1 = new City("М о с к в а",1.1e7);
c1.show();
```

Прототипное наследование

```
function Metropolis(name,population,location){
  City.apply(this,[name,population]);
  this.location = location;
}
Metropolis.prototype = new City();
Metropolis.prototype.constructor = Metropolis;
let c3 = new Metropolis("Е к а т е р и н б у р г",3e6);
c3.show();
console.log(c3.constructor);
```

- https://developer.mozilla.org/ru/docs/Web/JavaScript/Guide/Details_of_the_Object_Model

Расширение встроенных объектов

- Array, Boolean, Date, Error, Function, JSON, Math, Number, Object, RegExp
- не частая практика

```
let n = new Number(45);
let m = 45;

Number.prototype.currency = function(){
  return this + "р у б .";
}

console.log(n.currency());//"45 р у б "
console.log(m.currency());//"45 р у б "
```

Классы

```
class Название [extends Родитель] {
  constructor
  методы
}
```

Пример класса

```
class City{
  constructor(name,population){
    this.name = name;
    this.population = population;
  }
  show(){
    for(let i in this)
      console.log(`${i}=${this[i]}`);
  }
}
let c1 = new City("М о с к в а",1.1e7);
c1.show();
```

Наследование

```
class Metropolis extends City{
  constructor(name,population,location){
    super(name,population);
      this.location = location;
  }
}
let c2 = new Metropolis("Нижний Новгород",4e6,'коорд');
c2.show();
console.log(c2);
```

Геттеры и сеттеры

```
class City{
  constructor(name,population){
    this.name = name;
    this.population = population;
  }
  show(){
    for(let i in this)
      console.log(`${i}=${this[i]}`);
  }
  get cityName(){ return this.name; }
  set cityName(value){ this.name = value; }
}
let c1 = new City("Москва",1.1e7);
c1.cityName = "Новая Москва";
console.log(c1.cityName);
```

Практическая работа

1. Задание

```
/* Создать базовый класс Goods со свойствами
_title, _price;
 методом show() - показывающим св-ва объекта
 Создать экземпляр объекта (new Goods(...)) */
```

2. Задание

```
/* Создать класс наследник Phone со свойством
_type и значением "смартфон" и переопределить
конструктор Goods */
```

Статические свойства

```
class City{
  constructor(name,population){
    this.name = name;
    this.population = population;
  }
  show(){
    for(let i in this)
      console.log(`${i}=${this[i]}`);
  }
  get cityName(){ return this.name; }
  set cityName(value){ this.name = value; }

  static createCity(n,p){
    return new City(n,p);
  }
}
let c1 =  City.createCity("Москва",1.1e7);
```

Тест по наследованию и ООП

https://goo.gl/forms/nTv2V4nGdQCCtDvG2

Деструктуризация

```
let arr = [ "привет", 23 ];
let obj = { name: "Вася", age: 23 };

let [word,num] = arr;
let {name,age} = obj;
let {name:firstName,age:возраст} = obj;

console.log(word,num)
console.log(name,age)
console.log(firstName,возраст)
```

Пример деструктуризации с оператором spread

```
let arr = [ "п р и в е т", 23, 46, 100 ];

let [ ,num ] = arr;
let [ , , ...num2 ] = arr;

console.log(num);
console.log(num2);
```

Встроенные методы

- isFinite
- isNaN
- parseFloat
- parseInt

Number

- Number.MAX_VALUE Наибольшее число из возможных для представления
- Number.MIN_VALUE Наименьшее число из возможных для представления
- Number.NaN Специальное "Не числовое" ("not a number") значение
- parseInt()
- isFinite()
- new Number(5)

String

- length

- Методы:
- charAt, charCodeAt, **codePointAt** - по индексы символа находим...
- indexOf, lastIndexOf - по входному строковому аргументу находим позицию
- startsWith, endsWith, includes - проверяем наличие подстроки (вначале, в конце строки или просто наличие)
- concat - склейка/конкатенация строк
- fromCharCode, fromCodePoint - статические методы возвращают символ по коду
- split - по аргументу разбивает строку в массив
- slice - возвращает набор символов и одному или двум индексам (см. массивы)
- substring, substr - получает индекс и длину строки, которые нужно вернуть
- match, replace, search - поиск и замена, функции используются с регулярными

- toLowerCase, toUpperCase - получение строки в нижнем/верхнем регистре
- repeat(n) - возвращает строк повторяемую n раз
- trim() - удаляет все пробельные символы по краям строки

Пример indexOf

```
let str = `Непрерывная функция семантически.`
let s = "к";//речи

let i = 0;
while(1){
 i = str.indexOf(s,i);
 if( i != -1 )
  console.log(i++);
 else break;
}
```

Практическая работа

1. Задание

```
/* Есть строка:

let tut = `Умом Россию не понять,
Аршином общим не измерить:
У ней особенная стать -
В Россию можно только верить.`

Найти приблизительное кол-во слов в этой
строке
 */
```

2. Задание

```
/*Создать функцию substrCount(needle,haystack,offset,length)
которая находит количество вхождений
строки needle в строку haystack, со смещением offset
на длине length */
//...
let num = substrCount("ре","Ехал Грека через
реку",5,15); // 3
```

3. Задание /* Удалить каждый третий символ из строки "Ехал Грека через реку" */
4. Задание /* Удалить каждый символ Р до встречи символа З, а после З удалять Г в строке "Ехал Грека через реку" */

5. Задание

Date

- new Date()
- new Date("23 dec 2017")
- new Date(23423542423443);
- new Date(2020,2,12,23,10,7);

https://github.com/jquery/jquery-ui/blob/master/ui/i18n/datepicker-ru.js

Методы объекта Date

- "set" методы, служат для установки параметров объекта Date.
- "get" методы, служат для получения параметров объекта Date.
- "to" методы, служат для получения значения объекта Date в текстовом виде.
- "parse" и UTC методы, служат для распознавания дат и времени из текстового формата.

Пример работы с Date

- найти кол-во дней до ДР

```
/*Написать функцию, которая будет
определять кол-во дней, оставшихся до
ближайшего дня рождения 7 апреля */
let getDays = (month,day) => {
  month--;
 let  cdt = new Date();

 let year = (month == cdt.getMonth && day < cdt.getDate()) ?
   cdt.getFullYear()+1:cdt.getFullYear();

 let  hbd = new Date(year,month,day);
 let  sub = (hbd - cdt)/(1000*3600*24);
  return sub;
};
console.log(getDays(4,7));
console.log(getDays(12,25));
```

Практическая работа

1. Задание

```
/* Найти кол-во дней до НГ */
```

Error

- new Error()
- throw
- try/catch

Пример работы

```
class DivisionByZero extends Error{
  constructor(){
    super("Деление на ноль");
  }
}

let z = 0;
try{
  if(z == 0)
   throw new Error("Деление на ноль");
   //throw new DivisionByZero();
  console.log(5 / z);
}catch(er){
  console.log(er.message);
}
console.log("Дальше работаем..");
```

Пара слов о регулярных выражениях

https://codepen.io/anon/pen/OOpWBY?editors=1010