Объекты JavaScript

1. Тип данных - object

Объект в JS – ассоциативные массив (хэш). Хранит “свойства” любого типа «ключ : значение» и другие более сложные структуры и имеет ряд методов. ***Метод объекта*** – это просто функция, которая добавлена в ассоциативный массив. Данные объекта объединены общим смыслом, что позволяет оперировать объектом как единым целым.

Синтаксис для создания объекта «**student**» в ***JS***:

```
let student = new Object(); // конструктор объекта
let student = { };          // литерал объекта
```

```
let studentAdress = “student adress”;

let student = {
    name: “Ann”,
    age: 21,
    “student level”: “junior”,
    [studentAdress]: “Spb”, // «вычисляемое св-во»
    course: “JS dev”,
};
console.log(student);
```

Ключ состоит из нескольких слов с пробелом

Пары «ключ: значение» разделяются «,»

Для получения всего объекта необходимо обратится к переменной.

Объекты в JS – это ссылочный тип данных. При присваивании объекта переменной мы записываем в переменную не сам объект, а ссылку, указывающую на место в памяти, где он хранится.

```
. . .
alert (student.name);
. . .
alert ( student[“student level”]);
alert ( student[“studentAdress”]);
```

Обращение к отдельному свойству объекта можно через точку.

Можно через [ ].

2. Действия над объектами

Добавление свойств:

```
let student = {  // создаем объект
    name: “Ann”,
};
// добавляем свойства
student.age = 21;
student[“student level”] = “junior”;
student.course = “JS dev”;
```

Удаление свойств:

```
let student = {
    name: “Ann”,
    age: 21,
   “student level”: “junior”,
    course: “JS dev”,
};
// удаляем свойство
delete student.age;
```

Проверка наличия свойства:

```
let student = {
    name: “Ann”,
    age: 21,
   “student level”: “junior”,
    course: “JS dev”,
};
// проверка наличия св-ва “work experience”
if (student[“work experience”] === undefined){
  alert(‘Студент не имеет опыта работы’);
}
```

Изменение значения свойств:

```
let student = {
    name: “Ann”,
    age: 21,
   “student level”: “junior”,
    course: “JS dev”,
};
// изменяем имеющееся св-во
student.age = 29;
```

Вложенные объекты:

```
let student = {
    name: “Ann”,
    age: 21,
    skills: {
        html: true, css: true,
        react: false,
    },
};
Console.log(student.skills.css);
```

Перебор свойств объекта:

```
let student = {
    name: “Ann”,
    age: 21,
    skills: {
        html: true, css: true, react: false,
    },
};
for ( property in student.skills ) {
  console.log(property + “: ”+ student.skills[property]);
}
```

3. Действия над объектами

Копирование объекта:

```
let skills = {
    “1”: “React”,
    “2”: “HTML”,
    “3”: “JavaScript”,
};
let stillList = skills;
skillList[“4”] = “NodeJS”;
console.log(skills);
console.log(skillList);
```

Дублирование объекта:

```
let skills = {
    “1”: “React”,
    “2”: “HTML”,
    “3”: “JavaScript”,
};
let mySkills = Object.assign({ }, skills);
delete mySkills[“2”];
console.log(skills);
console.log(mySkills);
```

Добавление метода к объекту:

```
let rabbit = { };
// добавили новый метод к объекту
rabbit.run = function(n){
   alert(“Пробежал ”+ n + “ метров”);
};
rabbit.run(5);
rabbit.run(10);
```

```
let rabbit = {
  name: “John”,
  run(n) {  // сл. «function»- опущено
    alert(this.name);  // ==rabbit.name (при копировании будут проблемы !)
    alert(“Пробежал ”+ n + “ метров”);
  },
 };
rabbit.run(15);
```

4. Встроенные объекты JS

Язык JS предоставляет серию встроенных объектов.

5. Встроенные объекты JS

Global

Объект верхнего уровня, не имеет родителей. В нем определены свойства конструкторов для других встроенных объектов. Может расширятся свойствами и методами, которые при этом добавляются в «**глобальное пространство**».

Number

Служит для хранения целых чисел, для определения системных пределов **MAX_VALUE, MIN_VALUE**. Содержит методы **isNaN(), isFinite()**.

Boolean

Значения:
0, null, false, NaN, undefined, “” -> false;
Всё остальное -> true

String

Объект для работы со строками.
Методы:
replace() – замена одной части на другую,
charAt() – получение символа на позиции,
substring() – получение подстроки

RegExp

Проверка строк на соответствие заданному шаблону.
Методы:
test() – проверка строки на полное совпадение с шаблоном,
search() – поиск позиции совпадения строки и шаблона,
match() – получение строки найденного совпадения

```
let str = new String("QtScript");
let str2 = str.replace("Qt", "Java");      // str2="JavaScript
let c = str2.charAt(2);                    // c='v’
let str3 = str.substring(4, str.length);   // str2="Script"
```

```
let myRegexp = /[0-9]+/;
let myRegexp = new RegExp("[0-9]+");
alert(/[0-9]+/.test("number 95"));      // true
alert("number 95".search(/[0-9]+/));  // 7
alert("number 95".match(/[0-9]+/));  // "95"
```

6. Объект Array

Объект массива *Array* является контейнером, содержащим элементы данных. В *JS* не обязательно чтобы элементы массива были одного типа!

Создать массив можно посредством инициализации:

let arr = [“green”, “red”, “yellow”];

или

let arr = new Array (“green”, “red”, “yellow”);

Основные методы работы с массивами

Метод	описание
concat()	Вставка элемента в конец массива
join()	Объединение всех элементов в одну строку
pop()	Удаление последнего элемента массива
push()	Добавление элемента в конец массива
reverse ()	Изменение порядка элементов на обратный
shift ()	Удаление элементов в начале
slice ()	Возвращение подмножества массива
sort ()	Сортировка элементов
splice ()	Вставка и удаление элементов
toSource ()	Преобразование элементов массива в строку, заключенную в []
toString ()	Преобразование элементов в строку
unshift ()	Вставка элементов в начало массива
unwatch ()	Прекращение слежения за свойством
watch ()	Установка слежения за свойством

7. Основные методы объекта Array

Дополнение массива элементами.

- Динамическое дополнение элементами при ссылке на не существующий индекс:

```
let arr = [“red", “green", “blue“];
arr[3] = “black"; //=> arr= [ “red", “green", “blue", “black" ];
```

- Методом **push()** – в конец массива;
- Методом **unshift()** – в начало массива;
- Методом **splice()** – в середину (вставка);

```
arr.splice(1, 0, “pink”, “white”);
//=> arr= [ “red", “pink”, ”white”, “green", “blue", “black" ];
```

где 1-индекс начала вставки, 0 – кол-во символов которые нужно заменить при вставке, “pink”, “white”- список элементов вставки.

Преобразование массива в строку.

Метод **join()**, можно указывать разделитель:

```
let arr [“big", “medium”, “small“];
let str = arr.join(“**”);    //=> str=“big**medium**small”
```

Адресация элементов.

Доступ к элементам осуществляется посредством их имени (целые числа или строки):

```
let arr = new Array(2);
arr[“first”] = “John”;
arr[“second”] =  “Paul”;
let firstBeatle = arr[“first“];//=> firstBeatle=John
arr.second += “ McCartney”;
let secondBeatle = arr.second;
//=> secondBeatle = Paul McCart
```

Изменение порядка элементов массива.

Метод **reverse():**

```
let arr = [“first", “second", “third"];
let arr2 = arr.reverse(); //=> arr2=[“third", “second", “first”]
```

или просто:

```
let arr2 = [ “first”, “second", “third”] . reverse () ;
```

8. Основные методов объекта Array

Упорядочивание элементов массива.

Метод **sort()**:

```
let arr = [“red”, “blue", “green”];
let arr2 = arr.sort();
//=> arr2=[“blue”, “green”, “red”]
```

Объединение элементов массивов.

Метод **concat()**:

```
let arr = [“first”, “second”, “third”];
let arr2 = arr.concat([“fifth”, “sixth”]);
//=> arr2=[“first”, “second", “third”, “fifth”, sixth]
```

Многомерные массивы.

Создаются при помощи одномерных массивов:

```
let arr = [ [1, 1, 1], [2, 2, 2], [3, 3, 3] ]; // arr [3][3]
let n = arr[0][1]; //обращение к элементу arr (0,1) = 1;

let arr3 = [ [[1, 1, 1], [1, 1, 1], [1, 1, 1] ],
             [[2, 2, 2], [2, 2, 2], [2, 2, 2]],
             [[3, 3, 3] , [3, 3, 3] , [3, 3, 3]]
           ]; // arr3 [3][3][3]
let n = arr3[0][1][2]; //обращение к элементу arr3 (0,1,2) = 1;
```

9. Объект Date

Использовать даты до 1 января 1970 ГОДА ЗАПРЕЩЕНО!

Объект ***Date*** предназначен для хранения даты и времени (с точностью до млсек).
Для создания *объект Date* в его конструктор можно передать значение *года*, *месяца*, *дня*, *часа, минуты, секунды и миллисекунды* (эти параметры не обязательны). По умолчанию конструируется объект с **текущей** датой и временем.

```
let moment = new Date(2017, 2, 6, 23, 55, 30);
let now = new Date();
```

Основные методы:

- **getDay()** – номер дня недели (0-вс,…, 6-сб);
- **getDate()** – получение номера дня мес. (1 - 31);
- **getMonth()** – получение номера мес. (0 -11);
- **getFullYear()** – получение года вместе с тыс. (1900 + **getYear()**);
- **getHours ()** – получение значения часов;
- **getMinutes ()** - значение минут;
- **getSeconds ()** – значение секунд.

```
let month = ["January", "February", "March", "April", "Mai",
"Jun", "July", "August", "September", "October",
"November", "December”] ;
let now = new Date;
let strDateTime = now.getDate() + “. “
            + month[now.getMonth()] + “ ”
            + now.getFullYear()+ “ “
            + now.getHours()+ “:”
            + now.getMinutes()+ “:”
            + now.getSeconds();
asert(strDateTime);
```

-> “12. July 2024 16:47:25”

10. Объект Math

Объект содержит атрибуты и методы, используемые для проведения математических операций. Доступ к этому объекту можно получить **без использования конструктора**.
Все его атрибуты и методы определены как **статические.**

Атрибуты объекта Math

Константа	Значение
E	Число E. Значение константы Эйлера (2, 718281828459045)
LN2	Натуральный логарифм числа 2 (0,693147180559453)
LN10	Натуральный логарифм числа 10 (2,302585092994046)
LOG2E	Логарифм числа E по основанию 2 (1,442695040888963 3)
LOG10E	Логарифм числа E по основанию 10 (0,434294481903251 8)
PI	Число пи (3, 141592653589793)
SQRT1_2	Квадратный корень из числа 1/2 (0,707106781186547 6)
SQRT2	Квадратный корень из числа 2 (1,414213562373095 1)

11. Основные методы объекта Math

Модуль числа.

Метод **abs() –** абс.значение числа:

```
let x1 = Math.abs(-123.5);  // x1 = 123.5 ,  x2 = 123
let x2 = Math.abs(123);
```

Округление.

Методы **ceil(), floor(), round()**:

```
let x1 = Math.ceil(5.2);       //=> x1=6
let x2 = Math.ceil(-5.5);      //=> x2=-5
let x1 = Math.floor(-5.4);     //=> x1=-6
let x2 = Math.floor(5.9);      //=> x2=5
let x1 = Math.round(5.4);      //=> x1=5
let x2 = Math.round(5.5);      //=> x2=6
```

Вычисление квадратного корня.

Метод **sqrt()**:

```
let x = Math.sqrt(2); // => x= 1.41421356237309
let y = Math.sqrt(-3); // => y= NaN
```

Определение max и min.

Методы **max()**, **min()**:

```
let x1 = Math.max(5.4, 5.2);        //=> x1 = 5.4;
let x2 = Math.min(4, 3, 10, 2);     //=> x2 = 2;
```

Возведение в степень.

Метод **pow()**:

```
let x1 = Math.pow(2, 3); //=> x1= 8
```

Генератор случайных чисел.

Метод **random()**:

```
function randomize(range) {
    return (Math.random() * range)
}
 asert(randomize(1000));
```

12. Основные методы объекта Math

Тригонометрические методы.

Методы **cos(), sin(), tan(), asin(), acos(), atan()**:

```
let rad = -90 * Math.PI / 180;
asert (Math.sin(rad)); //=> -1

function ctg(angle) {
   return Math.cos(angle) / Math.sin (angle);
}

asert (Math.asin(-1) / Math.PI * 180); //=> -90
```

Вычисление натурального лагарифма.

Метод **log()**:

```
let x = Math.log(Math.E); // x = 1;

function log10(arg){
            return Math.log(arg) / Math.LN10;
}
let x = log10(10);  // => x=1
```

13. Объект Function

С его помощью можно использовать строку в качестве функции во время выполнения сценария. Для создания новой функции необходимо передать в конструктор параметры и текст.

Создадим свою функцию для объединения строк, принимающую в качестве аргументов четыре строки, причем последняя строка будет использоваться в качестве разделителя:

```
let myJoin = new Function(“a”, “b”, “c”, “sep”, “return a+ sep + b + sep +c”);

asert (myJoin ( “red”, “blue”, “green”, “**”));   //=> red**blue**green
```

*При создании объектов функций следует учитывать, что трансляция объекта ***Function*** осуществляется при каждом его использовании, вследствие чего выполняться такой код будет гораздо медленнее, чем при реализации обычных функций *JS*.

14. Домашка #4

// Что выведет следующий код?

```
let fruits = ["Яблоки", "Груша", "Апельсин"];

// добавляем новое значение в "копию"
let shoppingCart = fruits;
shoppingCart.push("Банан");

// что в fruits?
alert( fruits.length ); // ?
```

// Произведите 5 операций с массивом.

1. Создайте массив ***ягоды*** с элементами «**малина**» и «**клубника**».
2. Добавьте «**вишня**» в конец.
3. Замените значение в середине на «**арбуз**». Ваш код для поиска значения в середине должен работать для массивов с любой длиной.
4. Удалите первый элемент массива и покажите его.
5. Вставьте «**черника**» и «**земляника**» в начало массива.

// Создайте функцию getDateAgo(date, days), возвращающую число, которое было days дней назад от даты date.

```
let date = new Date(2015, 0, 2);

alert( getDateAgo(date, 1) );       // 1, (1 Jan 2015)
alert( getDateAgo(date, 2) );      // 31, (31 Dec 2014)
alert( getDateAgo(date, 365) );   // 2, (2 Jan 2014)
```