**JavaScript**

J.S. чувствителен к регистру.

JavaScript игнорирует пробелы, табуляции и переводы строк.

Идентификатор – это просто имя

0.1 + 0.2 == 0.3; // false

NaN - не числовое значение

DOM(Объектная Модель Документа) описывает структуру веб-станицы в виде древовидного представления и предоставляет доступ к отдельным элементам веб-станицы.

J.S. - интерпретируемый язык (читается по-строчно и тут же выполняется).

Компилируемый язык - преобр. в машинный язык и обратно, а затем выполняется.

Синхронная загрузка – поочерёдная загрузка.

Асинхронная загрузка – одновременная загрузка вместе с содержимым документа. Выполняется с помощью атрибута «async» и «defer»:

<script async src=””> - файл выведется как только загрузится.

<script defer src=””> - сработает, когда загрузится весь документ. Если файлов таких несколько первым сработает файл с именем, вышестоящим в алфавитном порядке.

-----------------------------------------Типы данных----------------

В JavaScript имеется пять *простых типов* данных:

* **number**: представляет числовое значение
* **string**: представляет строку
* **Boolean**: представляет логическое значение true или false
* **undefined**: указывает, что значение не установлено
* **null**: указывает на неопределенное значение

А также JavaScript имеет в свое арсенале дополнительные типы объектов, такие как:

Классы (множества объектов, определяемых конструктором «new»):

* 1. Array
  2. Fuction
  3. Date
  4. RegExp (регулярные выражения)
  5. Error (синтаксические ошибки и ошибки времени выполнения)

typeof (x) – возвращает тип переменной.

typeof {} //object

typeof null //object

typeof function() //function (тип данных: объект)

---------------------------------Преобразование типов------------------

1. Явное.

|  |  |  |
| --- | --- | --- |
| Number (null) // 0  Number (undefined) //NaN  Number (“ab12”) //12  Number (“ab”) //NaN  Number (false) //0  Number (true) //1 | Boolean (NaN) //false  Boolean (0) //false  Boolean (“false”) //true  Boolean (false) //false |  |

1. Неявное.

“ ”+1+0 //”10” (конкатенация)

“ “-1+0 //-1 (“-” не работает со строками)

6/”3” // 2

“2”\*”3” // 6

true + false //1

4+5+”px” //”9px” (4 и 5 не приводятся в строку так как они в начале)

“$”+4+5 //$45 (приводит в строку сначала 4, а потом 5)

“4”-2 // NaN

7/0 // Infinity

“-9/n”+5 // “-9/n5”

“-9/n”-5 // -14

5 && 2 // 2

2 && 5 //5

5 || 0 //5

0||5 //5

null + 1 //1

undefined + 1 //NaN

null == undefined

------------------------------Математические операции(объект Math())------------

Мат операции производятся с помощью свойств объкта Math:

Math.round() – округление до ближайшего целого

Math.ceil() – округление вверх

Math.floor() – округление вниз

Math.abs() – абсолютное значение

Math.max() – наибольший аргумент из предоставленных

Math.random() – случайное число в диапазоне (0; 1)

Math.sqrt() – корень квадратный

Math.pow(2, 1/3) – число в степени

-----------------------------------Даты (объект Date())---------------

var a = new Date(2017, 0, 5) - 5 января 2017 (месяцы считаются от «0»)

var a = new Date(2017, 0, 5, 17, 10, 30) – та же дата, в 17:10:30

var a = new Date( ) - текущая дата и время

a.getFullYear() - 2017

a.getMonth() - 0

a.getDate() - 5

a.getDay() – Пятница

a.getHours() – 17

-----------------Управляющие символы-----------------

\n – переход на другую строку

\t – горизонтальная табуляция

\v – вертикальная табуляция

\f – переход на другую страницу

\r – возврат каретки (на начало строки)

\” – двойная кавычка

\’ – одинарная кавычка

\\ - обратная коса черта \

\b – возврат на одну позицию (backspace)

--------------------------Работа со строками----------

var s = “Hello, world;

s[0] – первый символ строки (строки могут интерпретироваться как массивы, доступные только для чтения)

s[s.length-1] – последний символ строки

s.length – длина строки

s.slice() – символы в указанном интервале без указанных значений

s.indexOf() – символ по указанной позиции

s.split() – разбить строку на подстроки

s.replace() – заменить на другую строку.подстроку

-------------------Регулярные выражения (объект RegExp())-----------

/[1-9][0-9]\*/ - первый символ (1-9), второй (0-9), ( \*) - любое кол-во символов

/^dfc/ - символы в начале строки

/\dfsdfss\b/i - соответствует слову и учитывает регистр

/\d+/g - любое количество цифр

var t = “sds7d”; - образец текста

var p = “/\d+/g ”; - шаблон

p.test(t) - “true”, если совпало

t.search(p) - «3» (позиция первого совпадения)

-----------------------------Работа с массивами

## Вывод всех элементов массива через цикл

var arr = ["C#","C++","Java"];

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

{

document.write(arr[i] + "<br/>");

}

## Поиск индекса элемента

Методы **indexOf()** и **lastIndexOf()** возвращают индекс первого и последнего включения элемента в массиве.

var arr = ["C", "C#", "C++", "Java", "Ruby"];

var firstIndex = arr.indexOf("C");

var lastIndex = arr.lastIndexOf("Ruby");

var otherIndex = arr.indexOf("C++");

var notFoundIndex = arr.indexOf("Phyton");

document.write(firstIndex); // 0

document.write(lastIndex); // 4

document.write(otherIndex); // 2

document.write(notFoundIndex); // -1

Если указанного элемента в массиве нет, то возвращается значение -1.

## Метод push()

Метод push() добавляет элемент в конец массива:

var arr = [];

arr.push("C#");

arr.push("C++");

arr.push("Java");

console.log(arr)//C#,C++,Java

## pop()

Метод **pop()** извлекает и удаляет последний элемент из массива

var arr = ["C#", "C++", "Java"];

var lastItem = arr.pop(); // извлекаем из массива последний элемент

document.write(lastItem + "<br/>");//Java

console.log(arr); // C#,C++

## shift()

Метод **shift()** извлекает и удаляет элемент из начала массива:

var arr = ["C#", "C++", "Java"];

var firstItem = arr.shift();

document.write(firstItem + "<br/>");//C#

console.log(arr); //C++,Java

## unshift()

Метод **unshift()** добавляет новый элемент в начало массива:

var arr = ["C#", "C++", "Java"];

arr.unshift("C");

console.log(arr); //C,C#,C++,Java

## concat()

Метод concat() служит для объединения нескольких массивов в один результатирующий массив:

var arr = ["C#", "C++", "Java"];

var arr2 = ["HTML", "CSS", "JavaScript"];

var result = arr.concat(arr2);

console.log(result);// C#,C++,Java,HTML,CSS,JavaScript

## slice()

Метод slice() клонирует часть массива в новый массив. Первый параметр - индекс массива с которого начинается клонирование, второй - индекс элемента до которого происходит клонирование (не забываем, что индекс начинается с 0)

var arr = ["C", "C#", "C++", "Java", "Ruby"];

var cloneArr = arr.slice(1,3);

console.log(cloneArr);//C#,C++,Java

## join()

Метод **join()** объединяет все элементы массива в одну строку:

var arr = ["C#", "C++", "Java"];

var resultString = arr.join(" | ");

document.write(resultString);//C# | C++ | Java

Внутри скобок передаетс разделитель, с помощью которого будут разделены элементы массива в результатирующей строки. По умолчанию, используется (",")

-----------------------------trow

trow – оператор запускающий исключение (сигнализирование об ошибке)

try/catch/finally - обрабатывает исключение

-------------------------------Объекты--------

J.S. автоматически выполняет сборку мусора в память и удаление лишних объектов.

Объект – набор значений или др объектов.(Ассоциативный массив

Атрибуты свойств объектов:

writable – доступность для записи

enumerable – доступность для перечисления в цикле

configurable – возможность настройки (изменения)

Атрибуты объектов:

1. prototype – ссылка от др объекта, от которого наслед свойства
2. class – содержит класс объекта и тип объекта
3. extensible – возможность добавления новых свойств

Виды создания объектов:

1. var object = {…};
2. var object = new Object(); (прототипом является Object.prototype)
3. var x = Object.create(y); (“x” наследует свойства от “y”)

Цепочка прототипов: new Date() наследует свойства Date.prototype и Object.prototype

o.x – доступ к св-ву «х» объекта «о» (если св-ва нет – берет св-во прототипа – если нет – берет св-во прототипа прототипа и т.д.)

delete o.x - удаляет само св-во, а не только значение

Проверка существования св-ва: “x” in o; // true (если объект «о» имеет св-во «х»)

Перечисление св-в: for (p in o) {console.log(p)}

или o.keys()

Методы доступа в объекте вместо “function” имеют ключевые слова “get” и “set”:

var o = {

dt\_prop: value,

get ac\_prop () { },

set ac\_prop () { }

};

Объект Window представляет собой окно браузера и является родительским по отношению ко всем остальным объектам в языке JavaScript.

Методы объекта Window:

alert — отображает диалоговую панель Alert (Предупреждение) с сообщением и кнопкой OK;

open — открывает окно;

close — закрывает окно;

confirm — отображает диалоговую панель Confirm (Подтверждение) с кнопками OK и Отмена;

prompt — отображает диалоговую панель Prompt (Запрос) с полем ввода; setTimeout — устанавливает таймер;

clearTimeout — сбрасывает таймер.

Объект Navigator - с помощью которого можно получить неко- торые данные о браузере и компьютере пользователя. Методы и св-ва:

appVersion — определяет версию браузера, включая платформу, выпуск бра- узера и операционную систему пользователя.

cookieEnabled — определяет, разрешается или нет использование текста cookies

Объект Screen Объект Screen содержит информацию о различных параметрах экрана монитора пользователя.

Имеет следующие свойства: width — определяет полную ширину экрана монитора (в пикселах); height — определяет полную высоту экрана монитора (в пикселах);

document.write('width: '+window.screen.width+'');

Объект Location Этот объект содержит информацию об URL-адресе текущей страницы и его состав- ляющих.

href — полный URL-адрес текущей веб-страницы;

document.write(location.href)

Объект Document.

anchors — коллекция всех локальных меток («якорей»), размещенных в HTML- документе; эти метки содержатся в тегах и применя- ются для организации ссылок внутри одной веб-страницы;

links — коллекция всех ссылок в HTML-документе, содержащихся в тегах ; images — массив всех изображений на веб-странице;

forms — коллекция всех форм в документе HTML.

document.write(document.anchors.length)

getElementsByName({Имя\_элемента}) — возвращает элемент, специфическое имя которого передается в качестве параметра.

getElementById({ID})— возвращает элемент, имя которого передается в качестве параметра. Имя элемента страницы задается атрибутом ID. Един­ст- венный параметр этого метода — имя элемента страницы.

getElementsByTagName({Имя тега}) — возвращает тег, имя которого пе- редается в качестве параметра. write — записывает текст или HTML-код в текущее место документа.

Объект Form служит контейнером для пересылки данных от пользователя к сер- веру.

----------------------Массивы---------------

[1,2,3] == [1,2,3] // false (массив не равен массиву)

new Array (3) == “ , , ” // true

new Array(3).toString () ; // “ , , ”

new Array(3) === “ , , ”; // false

var empty = [ ]; var a = new Array();

Добавить значение в конец массива: a.push(“zero”);

Удалить значение массива с индексом «1»: delete a[1];

Обход элементов массива:

for (var i=0, I < a.length; i++) {

if (!a[i]) continue; //пропустить null, undefined

}

Методы класса Array:

1. .join - объед элементы в строку
2. .reverse - меняет порядок следования эл-в на обратный
3. .sort - сортирует элементы
4. .concat - добавляет элементы
5. .slice - возвр фрагмент массива
6. spice - вставка и удаление эл-в массива
7. .indexOf - возвр элемент массива с указан индексом
8. .push - добавляет эл в конец массива

------------------------------Функции-----------------------

Переменные, объявленные внутри функции, существуют только в пределах ф-ии.

Переменные, объявленные за пределами ф-ии, являются глобальными и видимы в любой точке программы.

Что бы получить ф-ию, возвращающее уникальное целое число при каждом своем вызове, в качестве переменной, хранящей это значение, используется свойсво объекта Function.

( function() { } ) – определение и вызов функции в одном выражении испоьзуется для инкапсуляции переменных, что бы предовратить конфликты с переменных других программами.

Пример области видимости:

var scope = "глобальная";

function f() {

alert(scope); // Показывает "undefined", а не "глобальная".

var scope = "локальная"; // Переменная инициализируется здесь,

// но определена она везде в функции.

alert(scope); // Показывает "локальная"

}

f();

------------------------Работа с формами-----------

<form id="demo" >

<input type="text" id="name">

<input type=”submit”>

</form>

<script>

document.getElementById('demo').onsubmit = function () {

alert ('Hello, ' + document.getElementById('name').value + '!');

return false; // не нужно отправлять на сервер

}

</script>

На jQuery:

<script>

$(document).ready(function() {

$('#demo').submit(function(event) {

alert('Hello, ' + $('#name').val() + '!');

event.preventDefault();

});

});

</script>