JavaScript

Введение

Синтаксис

Объявлять переменные можно через let или var, а константы через const

```
var x, y;          // объявление переменных
x = 5; y = 6;      // инициализация переменных
z = x + y;         // вычисление значений
```

Важно помнить:

- переменная, объявленная, но никогда не инициализированная будет иметь значение "undefined";
- при повторном объявлении переменной её значение не изменится

```
var carName = "Volvo";
var carName; // всё равно будет "Volvo"
```

Типы данных

- Примитивные:
  1) число (number)
  2) строка (string)
  3) логический (boolean)
  4) null
  5) undefined
- Сложные:
  1) Объект (object)
  2) Функции (function)

Типы в JavaScript – д

```
var x;          // x = undefined
x = 5;          // x - число
x = "John";     // x - строка
```

Числа и строки

Числа:

- 10.50
- 1001
- 123e5 (= 12300000)
- 123e-5 (= 0.00123)

Текст:

- “John Doe“
- ‘John Doe’

Объекты

```
var person = {
    firstName : "Иван",
    lastName  : "Петров",
    age       : 50,
    eyeColor  : "голубой"
};

var johnsAge = person.firstName + ". Возраст: " + person.age + " лет."
// Иван. Возраст: 50 лет.

var johnsAgeNew = person['firstName'] + '. Возраст: ' + person['age'] + ' лет.';
// Иван. Возраст: 50 лет.
```

Оператор typeof

```
typeof 2 // number
typeof 0.2 // number
typeof
typeof {
    firstName : "John",
    lastName  : "Doe",
    age       : 50,
    eyeColor  : "blue"
} // object

var car;
typeof car // undefined

typeof null // object
```

null и undefined

```
typeof undefined // undefined
typeof null // object

null === undefined // false
null == undefined // true
```

Операторы

Оператор	Описание
+	Сложение
-	Вычитание
*	Умножение
/	Деление
%	Остаток от деления
++	Инкремент
--	Декремент

```
var x, y, z;

x = 5;
y = 6;
z = '12';

var xy = x + y; // 11
var xz = x + z; // '512'
var xzNumber = x + +z; // 17
```

Сравнение переменных

Оператор	Описание
==	равно
===	равно по типу и значению
!=	не равно
!==	не равно по типу и по значению
>	больше
<	меньше
>=	больше или равно
<=	меньше или равно

Сравнение переменных

```
var x = 5;
var y = '5';
var z = 6;

var xy = x == y; // true
var xz = x == z; // false
var xyByType = x === y; // false
```

Массивы

- Инициализация:

```
var cars = ["Saab", "Volvo", "BMW"];

var carsNew = new Array("Saab", "Volvo", "BMW");
```

- Тип:

```
typeof ["Saab", "Volvo", "BMW"] // object
```

- Обращение к элементу:

```
var car = cars[0];
```

- Длина массива:

```
var arrayLength = cars.length;
```

- Цикл for:

```
var cars = ["BMW", "Volvo", "Saab", "Ford"];

for (i = 0; i < cars.length; i++) {
    console.log(cars[i]);
}
```

Методы массивов

- .pop()

```
var fruits = ["Банан", "Апельсин", "Яблоко", "Манго"];
fruits.pop(); // удаляет последний элемент массива ("Манго")
var fruit = fruits.pop(); // fruit = последний элемент массива ("Яблоко")
                          // Последний элемент УДАЛИТСЯ из массива
```

- .push()

```
var fruits = ["Банан", "Апельсин", "Яблоко", "Манго"];
fruits.push("Киви"); // добавляет новый элемент в массив
var fruit = fruits.push("Ананас"); // fruit = 6 (количество элементов в массиве)
                                   // Новый элемент добавится в массив
```

Методы массивов

- .shift()

```
var fruits = ["Банан", "Апельсин", "Яблоко", "Манго"];
fruits.shift(); // удаляет первый элемент массива ("Банан")
var fruit = fruits.shift(); // fruit = первый элемент массива ("Апельсин")
                            // Первый элемент УДАЛИТСЯ из массива
```

- .unshift()

```
var fruits = ["Банан", "Апельсин", "Яблоко", "Манго"];
fruits.unshift("Мандарин"); // добавляет элемент ("Мандарин") в начало массива
var fruitCount = fruits.unshift("Киви"); // fruitCount = 6 (количество элементов в массиве)
                                         // Новый элемент добавится в начало массива
```

Функции

```
function myFunction(a, b){
    return a * b;
}
```

или

```
var myFunction = new Function("a", "b", "return a * b");
```

- типы параметров и возвращаемого значения не указываются;
- возвращать значение необязательно;
- Значение параметров по умолчанию – undefined;
- typeof myFunction // function;
- Object-параметры передаются по ссылке.

Вызов функций

- По событию (onclick, onload, …)

```
<button onclick="changeDivColor()">Сменить цвет!</button>
```

- Из JavaScript кода (из другой функции)

```
var result = myFunction(2, 3); // 4
```

- Автоматический (самовызывающиеся функции)

```
(function () {document.getElementById("helloWorld").innerHTML = "Hello world!"})();
```

Методы объектов

```
var myObject = {
    firstName: 'Иван',
    lastName: 'Петров',
    fullName: function() {
        return this.firstName + ' ' + this.lastName;
    }
}

myObject.fullName(); // Вернёт "Иван Петров"
```

Способы вставки JS

- в теге элемента:

```
<button onclick="document.getElementById('helloWorld').innerHtml ='HelloWorld'">
```

- в <head> и <body> (через тег <script>)
- в отдельном *.js файле (через тег <script>)

```
<script src="myScript.js"></script>
```

События

События мыши:

- `click` – происходит, когда кликнули на элемент левой кнопкой мыши
- `contextmenu` – происходит, когда кликнули на элемент правой кнопкой мыши
- `mouseover` – возникает, когда на элемент наводится мышь
- `mousedown` и `mouseup` – когда кнопку мыши нажали или отжали
- `mousemove` – при движении мыши

События на элементах управления:

- `submit` – посетитель отправил форму `<form>`
- `focus` – посетитель фокусируется на элементе, например нажимает на `<input>`

Клавиатурные события:

- `keydown` – когда посетитель нажимает клавишу
- `keyup` – когда посетитель отпускает клавишу

События документа:

- `DOMContentLoaded` – когда HTML загружен и обработан, DOM документа полностью построен и доступен.

События CSS:

- `transitionend` – когда CSS-анимация завершена.

Загрузка документа

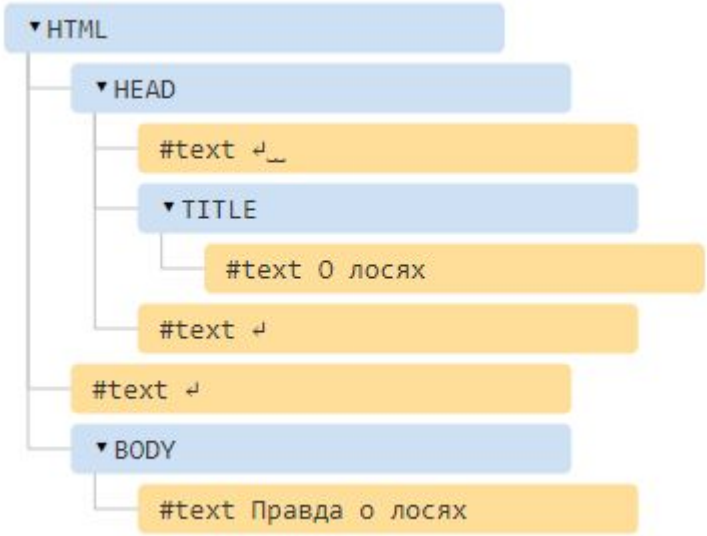
- DOMContentLoaded – браузер полностью загрузил HTML и построил DOM-дерево
- load – браузер загрузил все ресурсы
- beforeunload / unload – уход со страницы

Объект document

- document – объект HTML-страницы. По сути, это корневой DOM-узел
- Содержит методы для работы с нодой

- DOM

```
<!DOCTYPE HTML>
<html>
<head>
  <title>О лосях</title>
</head>
<body>
  Правда о лосях
</body>
</html>
```



getElementById()

Если элементу назначен специальный атрибут id, то можно получить его прямо по переменной с именем из значения

Пусто!

document.getElementById(“helloWorld").innerHTML = "Hello JavaScript";

 Hello JavaScript

querySelector(css), querySelectorAll(css)

Вызов querySelectorAll возвращает все элементы внутри элемента, которому соответствует селектор

```
<div>
  <p>1</p>
  <p>2</p>
</div>

var elements = document.querySelectorAll('div’);

  for (var i = 0; i < elements.length; i++) {
    alert( elements[i].innerHTML ); // 1, 2
  }
```

getElementsByTagName(tag)

- Метод getElementsByTagName ищет все элементы с заданным тегом внутри элемента и возвращает их в виде списка

```
var tableElem = document.getElementById('age-table');
 var elements = tableElem.getElementsByTagName('input');

  for (var i = 0; i < elements.length; i++) {
    var input = elements[i];
    alert( input.value + ': ' + input.checked );
  }
```

Справочная информация

Метод	Ищет по...	Ищет внутри элемента?	Поддержка
`getElementById`	`id`	-	везде
`getElementsByName`	`name`	-	везде
`getElementsByTagName`	тег или `'*'`	✓	везде
`getElementsByClassName`	классу	✓	кроме IE8-
`querySelector`	CSS-селектор	✓	везде
`querySelectorAll`	CSS-селектор	✓	везде

Вопросы?