**Javascript**

Подключение кода javascript на html-страницу осуществляется с помощью тега <script>. Данный тег следует размещать либо в заголовке (между тегами <head> и </head>), либо в теле веб-страницы (между тегами <body> и </body>). Нередко подключение скриптов происходит перед закрывающим тегом </body> для оптимизации загрузки веб-страницы.

Когда браузер получает веб-страницу с кодом html и javascript, то он ее интерпретирует. Результат интерпретации в виде различных элементов - кнопок, полей ввода, текстовых блоков и т.д., мы видим перед собой в браузере. Интерпретация веб-страницы происходит последовательно сверху вниз.

Когда браузер встречает на веб-странице элемент <script> с кодом javascript, то вступает в действие встроенный интерпретатор javascript. И пока он не закончит свою работу, дальше интерпретация веб-страницы не идет.

Рассмотрим пример

<!DOCTYPE html>

<html>

<head>

    <meta charset="utf-8" />

    <title>JavaScript</title>

    <script>

    alert("Секция head");

    </script>

</head>

<body>

    <h2>Первый заголовок</h2>

    <script>

    alert("Первый заголовок");

    </script>

    <h2>Второй заголовок</h2>

    <script>

    alert("Второй заголовок");

    </script>

</body>

</html>

Здесь три вставки кода javascript - один в секции <head> и по одному после каждого заголовка.

Откроем веб-страницу в браузере.

Несмотря на то, что на веб-странице объявлено два заголовка, но в браузере их еще не видно, поскольку выполняется код javascript из секции head. И пока не закроем окно сообщения, дальше интерпретация веб-станицы идти не будет.

После закрытия окна с сообщением браузер добавит на страницу первый заголовок и снова остановится на следующем блоке кода javascript.

После закрытия окна сообщения браузер двинется дальше и добавит на страницу второй заголовок и остановится на третьем блоке кода javascript.

После закрытия третьего окна с сообщением браузер закончит интерпретацию веб-страницы, и веб-страница окажется полностью загружена.

Для увеличения производительности нередко вставки кода javascript идут перед закрывающим тегом **</body>**, когда основная часть веб-страницы уже загружена в браузере.

Рассмотрим пример

<!DOCTYPE html>

<html>

<head>

    <meta charset="utf-8" />

    <title>JavaScript</title>

</head>

<body>

    <h2>Первый заголовок</h2>

   <h2>Второй заголовок</h2>

    <script>

    alert("Секция head");

    alert("Первый заголовок");

    alert("Второй заголовок");

    </script>

</body>

</html>

**Основы синтаксиса javascript**

Рекомендуется определять каждую инструкцию javascript на отдельной строчке и завершать ее точкой с запятой.

Комментарии могут быть однострочными, для которых используется двойной слеш

// вывод сообщения

alert("Вычисление выражения");

// арифметическая операция

var a = 5 + 8;

alert(a);

Кроме однострочных комментариев могут использоваться и многострочные. Такие комментарии заключаются между символами /\*текст комментария\*/

/\*  вывод сообщения и

 арифметическая операция \*/

alert("Вычисление выражения");

var a = 5 + 8;

alert(a);

**Подключение внешнего файла JavaScript**

Еще один способ подключения кода JavaScript на веб-страницу представляет вынесение кода во внешние файлы и их подключение с помощью тега <script>

Создадим в каталоге **app** новый подкаталог  **js**. Он будет предназначен для хранения файлов с кодом javascript. В этом подкаталоге создадим новый текстовый файл, который назовем **myscript.js**

var date = new Date(); // получаем текущую дату

var time = date.getHours(); // получаем текущее время в часах

if(time < 13) // сравниваем время с число 13

   alert('Доброе утро!'); // если время меньше 13

else

    alert('Добрый вечер!'); // если время равно 13 и больше

Подключим в коде на HTML код на Javascript

<!DOCTYPE html>

<html>

<head>

    <meta charset="utf-8" />

<title>JavaScript</title>

</head>

<body>

    <h2>Первая программа на JavaScript</h2>

    <script src="js/myscript.js"></script>

</body>

</html>

**Консоль браузера, console.log и document.write**

Консоль браузерапозволяет производить отладку программы. Можно напрямую вводить в консоль браузера выражения JavaScript, и они будут выполняться.

Для вывода различного рода информации в консоли браузера используется специальная функция **console.log()**.

Рассмотрим пример

<!DOCTYPE html>

<html>

<head>

    <meta charset="utf-8" />

<title>JavaScript</title>

</head>

<body>

    <h2>Первая программа на JavaScript</h2>

<script>

    var a = 5 + 8;

    console.log("Результат операции");

    console.log(a);

    </script>

</body>

</html>

На начальном этапе может быть полезен метод **document.write()**, который пишет информацию на веб-страницу.

<!DOCTYPE html>

<html>

<head>

    <meta charset="utf-8" />

<title>JavaScript</title>

</head>

<body>

    <h2>Первая программа на JavaScript</h2>

    <script>

    var a = 5 + 8;

    document.write("Результат операции ");

    document.write(a);

    </script>

</body>

</html>

**Основы javascript**

Каждая переменная имеет имя. Имя представляет собой произвольный набор алфавитно-цифровых символов, знака подчеркивания (\_) или знака доллара ($), причем названия не должны начинаться с цифровых символов.

Нельзя давать переменным такие имена, которые совпадают с зарезервированными ключевыми словами.

Список зарезервированных слов в JavaScript:

**abstract, boolean, break, byte, case, catch, char, class, const, continue, debugger, default, delete, do, double, else, enum, export, extends, false, final, finally, float, for, function, goto, if, implements, import, in, instanceof, int, inteface, long, native, new, null, package, private, protected, public, return, short, static, super, switch, synchronized, this, throw, throws, transient, true, try, typeof, var, volatile, void, while, with**

При названии переменных надо учитывать, что JavaScript является **регистрозависимым** языком.

**Типы данных**

В JavaScript имеется пять примитивных типов данных:

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

Все данные, которые не попадают под вышеперечисленные пять типов, относятся к типу **object**

**Числовые данные**

Числа в JavaScript могут иметь две формы:

Целые числа, например, 35. Мы можем использовать как положительные, так и отрицательные числа. Диапазон используемых чисел: от -253 до 253

Дробные числа (числа с плавающей точкой), например, 3.5575. Опять же можно использовать как положительные, так и отрицательные числа. Для чисел с плавающейточкой используется тот же диапазон: от -253 до 253

**Строки**

Тип string представляет строки, то есть такие данные, которые заключены в кавычки. Например, "Привет мир". Причем можно использовать как двойные, так и одинарные кавычки: "Привет мир" и 'Привет мир'. Единственное ограничение: тип закрывающей кавычки должен быть тот же, что и тип открывающей, то есть либо обе двойные, либо обе одинарные.

Если внутри строки встречаются кавычки, то мы их должны экранировать слешем. Например, пусть есть текст "Бюро "Рога и копыта"". Теперь экранируем кавычки:

var companyName = "Бюро \"Рога и копыта\"";

Также можно внутри строки использовать другой тип кавычек:

var companyName1 = "Бюро 'Рога и копыта'";

var companyName2 = 'Бюро "Рога и копыта"';

**Тип Boolean**

Тип Boolean представляет булевые или логические значения true и false (то есть да или нет):

var isAlive = true;

var isDead = false;

**null и undefined**

Нередко возникает путаница между null и undefined. Итак, когда мы только определяем переменную без присвоения ей начального значения, она представляет тип undefined:

var isAlive;

console.log(isAlive); // выведет undefined

Присвоение значение null означает, что переменная имеет некоторое неопределенное значение (не число, не строка, не логическое значение), но все-таки имеет значение (undefined означает, что переменная не имеет значения)

var isAlive;

console.log(isAlive); // undefined

isAlive = null;

console.log(isAlive); // null

**Слабая типизация**

JavaScript является языком со слабой типизацией. Это значит, что переменные могут динамически менять тип. Например:

var xNumber;  // тип undefined

console.log(xNumber);

xNumber = 45;  // тип number

console.log(xNumber);

xNumber = "45"; // тип string

console.log(xNumber);

var xNumber = 45;  // тип number

var yNumber = xNumber + 5;

console.log(yNumber); // 50

xNumber = "45"; // тип string

var zNumber = xNumber + 5

console.log(zNumber); // 455

**Оператор typeof**

С помощью оператора **typeof** можно получить тип переменной:

var name = "Tom";

console.log(typeof name); // string

var income = 45.8;

console.log(typeof income); // number

var isEnabled = true;

console.log(typeof isEnabled); // boolean

var undefVariable;

console.log(typeof undefVariable); // undefined

**Операции с переменными**

**Сложение**

var x = 10;

var y = x + 50;

**Вычитание**

var x = 100;

var y = x - 50;

**Умножение**

var x = 4;

var y = x \* 5;

**Деление**

var x = 40;

var y = x / 5;

**Деление по модулю** (оператор %) возвращает остаток от деления:

var x = 40;

var y = 7;

var z = x % y;

console.log(z); // 5

**Инкремент**:

// префиксный инкремент

var x = 5;

var z = ++x;

console.log(x); // 6

console.log(z); // 6

// постфиксный инкремент

var a = 5;

var b = a++;

console.log(a); // 6

console.log(b); // 5

**Декремент**

// префиксный декремент

var x = 5;

var z = --x;

console.log(x); // 4

console.log(z); // 4

// постфиксный декремент

var a = 5;

var b = a--;

console.log(a); // 4

console.log(b); // 5

**Операции присваивания**

**=, +=, -=, \*=, /=, %=**

**Операторы сравнения**

**==, ===, !=, <=, <, >=, >**

**Логические операции**

**&&, ||, !**

**Операции со строками**

**+**

**Преобразования данных**

**parseInt()**

var number1 = "46";

var number2 = "4";

var result = parseInt(number1) + parseInt(number2);

console.log(result); // 50

**parseFloat()**

var number1 = "46.07";

var number2 = "4.98";

var result = parseFloat(number1) + parseFloat(number2);

console.log(result); //51.05

**NaN** (Not a Number)

**isNaN()**

var num1 = "javascript";

var num2 = "22";var result = isNaN(num1);

console.log(result); // true - num1 не является числом

result = isNaN(num2);

console.log(result); //  false - num2 - это число

var myArray = [];

var people = ["Tom", "Alice", "Sam"];

console.log(people);

console.log(people[7]); // undefined

**Многомерные массивы**

var people = [

        ["Tom", 25, false],

        ["Bill", 38, true],

        ["Alice", 21, false]

];

console.log(people[0]); // ["Tom", 25, false]

console.log(people[1]); // ["Bill", 38, true]

var tomInfo = people[0];

console.log("Имя: " + people[0][0]); // Tom

console.log("Возраст: " + people[0][1]); // 25

var numbers = [];

numbers[0] = []; // теперь numbers - двумерный массив

numbers[0][0]=[]; // теперь numbers - трехмерный массив

numbers[0][0][0] = 5; // первый элемент трехмерного массива равен 5

console.log(numbers[0][0][0]);

**Выражение if**

if(условие) действия;

var income = 100;

var age = 19;

if(income < 150 && age > 18){

   var message = "доход больше 50";

    alert(message);

}

var myVar = 89;

if(myVar){

    // действия

}

if (typeof myVar != "undefined") {

   // действия

}

var age = 17;

if(age >= 18){

    alert("Вы допущены к программе кредитования");

}

else{

    alert("Вы не можете участвовать в программе, так как возраст меньше 18");

}

var income = 300;

if(income < 200){

    alert("Доход ниже среднего");

}

else if(income >= 200 && income <= 400){

    alert("Средний доход");

}

else{

    alert("Доход выше среднего");

}

**True или false**

**undefined**

Возвращает false

**null**

Возвращает false

**Boolean**

Если переменная равна false, то возвращается false. Соответствено если переменная равна true, то возвращается true

**Number**

Возвращает false, если число равно 0 или NaN (Not a Number), в остальных случаях возвращается true

**String**

Возвращает false, если переменная равна пустой строке, то есть ее длина равна 0, в остальных случаях возвращается true

**Object**

Всегда возвращает true

console.log(undefined==null); // true

**Конструкция switch..case**

var income = 300;

switch(income){

   case 100 :

        console.log("Доход равен 100");

        break;

    case 200 :

       console.log("Доход равен 200");

       break;

    case 300 :

        console.log("Доход равен 300");

        break;

}

var income = 300;

switch(income){

    case 100 :

        console.log("Доход равен 100");

        break;

    case 200 :

        console.log("Доход равен 200");

        break;

    case 300 :

        console.log("Доход равен 300");

        break;

    default:

        console.log("Доход неизвестной величины");

        break;

}

**Тернарная операция**

var a = 1;

var b = 2;

var result = a < b ? a + b : a - b;

console.log(result); // 3

* **for**
* **for..in**
* **while**
* **do..while**

**Цикл for**

for ([инициализация счетчика]; [условие]; [изменение счетчика]){

    // действия

}

var people = ["Tom", "Alice", "Bob", "Sam"];

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

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

}

var people = ["Tom", "Alice", "Bob", "Sam"];

for(var i = people.length - 1; i >= 0; i--){

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

}

**for..in**

Цикл for..in предназначен для перебора массивов.

for (индекс in массив) {

    // действия

}

var people = ["Tom", "Alice", "Bob", "Sam"];

for(var index in people){

    document.write(people[index] + "</br>");

}

**Цикл while**

Цикл while выполняется до тех пор, пока некоторое условие истинно.

while(условие){

    // действия

}

var people = ["Tom", "Alice", "Bob", "Sam"];

var index = 0;

while(index < people.length){

    document.write(people[index] + "</br>");

    index++;

}

**do..while**

В цикле do сначала выполняется код цикла, а потом происходит проверка условия в инструкции while. И пока это условие истинно, цикл повторяется.

var x = 1;

do{

    document.write(x \* x + "</br>");

    x++;

}while(x < 10)

### Операторы continue и break

# Функциональное программирование

## Функции

Синтаксис определения функции:

function имя\_функции([параметр [, ...]]){

     // Инструкции

}

Можно использовать анонимные функции:

var display = function(){  // определение функции

     document.write("функция в JavaScript");

}

display();

**Параметры функции**

### Результат функции

Для возвращения результата используется **return**