**Лабораторная работа №1**

*Операторы ветвлений и логические условия в JavaScript.*

*Типы переменных и доступные с ними методы и свойства.*

*Операторы циклов, switch, рекурсии, стеки в JavaScript*

**Задание 1**

**Вариант 1**: Напишите код, который получает значение двух переменных a и b через prompt, а затем выводит alert, затем перепишите следующий if с использованием оператора '?':

*if (sqrt(a\*b) < 4) {*

*result = 'Мало';*

*} else {*

*result = 'Много';*

*}*

**Вариант 2**: Напишите код, который получает значение двух переменных a и b через prompt, а затем выводит alert, затем перепишите следующий if с использованием оператора '?':

*if (a==b) {*

*result = 'значения равны';*

*} else {*

*result = 'значения не равны';*

*}*

**Вариант 3**: Напишите код, который получает значение трех переменных a, b и с через prompt, а затем выводит alert, затем перепишите следующий if с использованием оператора '?':

*if ((a\*a+b\*b)=с\*c) {*

*result = 'Ура: угадали длины сторон треугольника';*

*} else {*

*result = 'Надо учить теорему';*

*}*

**Вариант 4**: Напишите код, который получает значение двух переменных a и b через prompt, а затем выводит alert, затем перепишите следующий if с использованием оператора '?':

*if ((a+a)==b) {*

*result = 'значения равны';*

*} else {*

*result = 'значения не равны';*

*}*

**Задание 2**

**Вариант 1**: Напишите код, который получает значение возраста и пола (с проверкой условия, что вводится только «м» или «ж»). Напишите условие if для проверки того факта, что переменная с возрастом находится между 14 и 25 включительно и если пол мужской, то выводить фраза «Добрый день, молодой человек», если тот же возраст, но пол женский, то выводить фразу «Приветствуем юную даму». Во всех остальных случаях выводить «Привет!».

**Вариант 2**: Напишите код, который получает должность (с проверкой, что должность введена «начальник», «зам.начальника» или «секретарь») и ФИО. Напишите условие if, если должность не «секретарь», то выводить «Добрый день, господин начальник», иначе если должность «секретарь», то выводить «Добрый день, ФИО».

**Вариант 3**: Напишите код, который получает значения для трех переменных: цвет, форма, размер. При этом предлагается пользователю в сообщении вводить для цвета - «синий» или «зеленый» или «красный», для формы - «круглый», «квадратный», для размера – «большой», «маленький». Напишите условие, если цвет не синий, а форма круглая и размер маленький, то выводить фразу «Наверное вы подумали про яблоко», если форма квадратная, а цвет любой, то выводить фразу «Наверное вы подумали про нечто квадратное», иначе «Понятия не имею, о чем вы думали».

**Вариант 4**: Напишите код, который получает значения для трех переменных: цвет, форма, размер. При этом предлагается пользователю в сообщении вводить для цвета - «синий» или «зеленый» или «красный», для формы - «круглый», «квадратный», для размера – «большой», «маленький». Напишите условие, если цвет не красный и не зеленый, а форма круглая и размер большой, то выводить фразу «Что-то синее и большое», если форма квадратная, а цвет не синий, то выводить фразу «Наверное вы подумали про яблоко, но почему квадратное?», иначе выводить значения переменных в одном сообщении.

**Задание 3**

**Вариант 1**: Написать скрипт, в котором будет предлагаться ввести текст размером не менее случайного сгенерированного программного числа (генерировать в диапазоне от 10 до 100) символов. Определить количество символов во введенном тексте, наличие вхождений таких стоп-слов, как «был», «имеет», «есть». Вывести общим сообщением информацию о длине текста и вхождениях указанных слов.

**Вариант 2**: Написать скрипт, в котором будет предлагаться ввести текст размером не менее случайного сгенерированного программно числа (генерировать в диапазоне от 10 до 100) символов. Затем ввести текст размером в три раза меньше предыдущего. Если второй введенный текст встречается в первом введенном тексте, то вывести позицию вхождения второй строки в первую. Иначе сообщить, что совпадения нет.

**Вариант 3**: Написать скрипт, в котором будет предлагаться ввести текст размером не менее случайного сгенерированного программно числа (генерировать в диапазоне от 0 до 50) символов. Поменять местами в строке первое и последнее слово и вывести в сообщении полученный результат.

**Вариант 4**: Написать скрипт, в котором будет предлагаться ввести текст размером не менее случайного сгенерированного программно числа (генерировать в диапазоне от 1 до 45) символов. Все четные символы перевести в верхний регистр, все нечетные в нижний регистр. Вывести в сообщении полученный результат.

**Задание 4**

**Вариант 1**: Напишите цикл, который предлагает через prompt ввести один из вариантов предлагаемого текста (например: укажите пол: мужской/женский ). Если посетитель ввел другое значение, попросить ввести еще раз, и так далее. Цикл должен спрашивать, пока посетитель не введет нужное значение, либо не нажмет кнопку Cancel (ESC).

**Вариант 2**: Напишите цикл, который предлагает через prompt ввести число (в диапазоне от 15 до 90) и некоторый текст. Если посетитель ввел другое число или текст длиной меньше 10, то попросить ввести еще раз, и так далее. Цикл должен спрашивать число и текст пока либо посетитель не введет требуемые данные, либо не нажмет кнопку Cancel (ESC).

**Вариант 3**: Напишите цикл, который предлагает через prompt ввести число. Умножить в цикле число на случайное число. Если полученное число меньше 100, попросить ввести еще раз, и так далее.

**Вариант 4**: Напишите цикл, который предлагает через prompt ввести число больше 100 и некоторый текст. Если посетитель ввел другое число, то попросить ввести еще раз, и так далее. Цикл должен спрашивать число, пока посетитель не введет требуемые данные, либо не нажмет кнопку Cancel (ESC). При выходе из цикла вывести второе слово из введенного текста или сообщить, что текст состоит из одного слова.

**Задание 5**

**Вариант 1**: Создайте скрипт, который выводит все простые числа из интервала от 2 до 100. В результате полученные выводить числа одной строкой через запятую.

**Вариант 2**: Создайте скрипт, который выводит все числа из интервала от 2 до 100, которые делятся на 3 и 17 с остатком 1. В результате полученные выводить числа одной строкой через запятую.

**Вариант 3**: Создайте скрипт, который выводит все числа из интервала от 2 до 100, которые делятся на 5 и 11 с остатком 2. В результате полученные выводить числа одной строкой через запятую.

**Вариант 4**: Создайте скрипт, который выводит все числа из интервала от 2 до 100, которые делятся на 13 и 7 с остатком 3. В результате полученные выводить числа одной строкой через запятую.

**Задание 6**

**Вариант 1**: Написать скрипт, который будет запрашивать у пользователя ввод даты. Далее использовать оператор switch, который будет для 1,2 и 12 месяцев выводить в сообщении «зима», для 3,4,5 месяцев выводить «весна», для 6,7,8 месяцев выводить «лето», для 9,10,11 месяцев – «осень».

**Вариант 2**: Написать скрипт, который будет запрашивать у пользователя ввод времени в формате HH:MI (часы:минуты). Далее использовать оператор switch, который будет для времени с 07:00 до 12:00 выводить в сообщении «утро», с 12:00 до 17:00 выводить «день», c 17:00 до 21:00 выводить «вечер», с 21:00 до 07:00– «ночь».

**Вариант 3**: Написать скрипт, который будет запрашивать у пользователя ввод сведений о погоде (дождливо, ветренно, солнечно, снежно и т.п.). Далее использовать оператор switch, который будет для «дождливо» выводить в сообщении «захватите зонт и оденьте калоши», для «ветренно» выводить «укутайтесь потеплее», для «солнечно» выводить «возьмите солнечные очки-берегите глаза», для «снежно» – «пора лепить снеговиков», во всех остальных вариантов выводить «одевайтесь, как хотите, но погода непредсказуема».

**Вариант 4**: Написать скрипт, который будет запрашивать у пользователя ввод признаков описания грибов по цвету (белый, желтый, красный, коричневый), и наличию юбки (есть, нет). Далее использовать оператор switch, который будет для белого гриба без юбки выводить в сообщении «вы нашли белый гриб», для красного гриба с юбкой выводить «а не мухомор ли это», для красного гриба без юбки выводить «может это подосиновик», для остальных выводить «с грибами надо быть внимательными».

**Отчет по лабораторной работе**

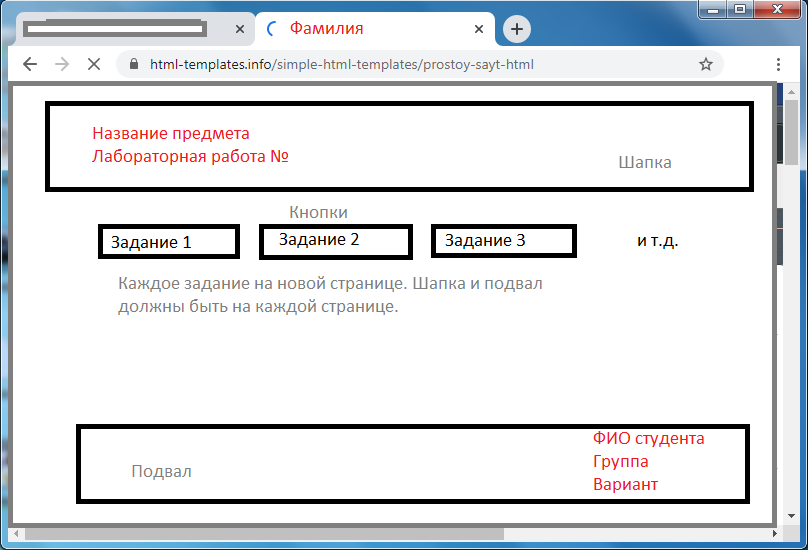
В соответствии со структурой заготовки отчета и примером оформления оформить в отчете все задания, выполняемые в ходе лабораторной работы, а также индивидуальные задания по вариантам. Файл с отчетом называть по шаблону: Фамилия\_лаб\_раб\_номер.

Отчет предоставляется в электронном виде либо лично преподавателю, либо на электронную почту для проверки. Также по результатам лабораторной работы на следующем за ней занятии проводится выборочный опрос по командам языка.

Пример страницы:

Должен быть фон страницы, использованы стили.

Стилизация у каждого студента должна быть разной. Если стилизация повторяется, она будет засчитана студенту, который сдаст ЛР раньше.



**Теория и примеры**

**Оператор if**

Оператор if («если») получает условие. Он вычисляет его, и если результат — true, то выполняет команду.

Если нужно выполнить более одной команды при выполнении условия, то они оформляются блоком кода в фигурных скобках:

*var year = prompt('В каком году появилась спецификация ECMA-262 5.1?', '');*

*/\* функция prompt создает окно для ввода текста, который передается в*

*переменную*

*\*/*

*if (year != 2011)*

*{*

*alert('А вот..');*

*alert('..и неправильно!');*

*}*

**Преобразование к логическому типу**

Оператор if вычисляет и преобразует условие к логическому типу.

!!! В логическом контексте число 0, пустая строка "", null и undefined, а также NaN являются false, остальные значения — true. Например, такое условие никогда не выполнится:

*if (0) { // 0 преобразуется к false*

*}*

*А такое — выполнится всегда:*

*if (1) { // 1 преобразуется к true*

*}*

Вычисление условия в проверке if (year != 2011) может быть вынесено в отдельную переменную:

*var cond = (year != 2011); // вернет true или false в зависимости от year*

*if (cond) {*

*//какие-то операторы*

*}*

Пример кода в составе страницы HTML:

*<!DOCTYPE HTML>*

*<html>*

*<head>*

*<title>Пример 2\_1</title>*

*</head>*

*<body>*

*<script>*

*var year = prompt('В каком году появилась спецификация ECMA-262 5.1?', '');*

*if (year != 2011)*

*{*

*alert('А вот..');*

*alert('..и неправильно!');*

*}*

*</script>*

*</body>*

*</html>*

**Условие else**

Необязательный блок else («иначе») выполняется, если условие неверно:

*var year = prompt('Введите год выхода ECMA-262 5.1', '');*

*if (year == 2011) {*

*alert('Да вы знаток!');*

*} else {*

*alert('А вот и неправильно!'); // любое значение, кроме 2011*

*}*

**Несколько условий, else if**

Бывает нужно проверить несколько вариантов условия. Для этого используется блок else if:

*var year = prompt('В каком году появилась спецификация ECMA-262 5.1?', '');*

*if (year < 2011) {*

*alert('Это слишком рано..');*

*} else if (year > 2011) {*

*alert('Это поздновато..');*

*} else {*

*alert('Да, точно в этом году!');*

*}*

Пример кода в составе страницы HTML:

*<!DOCTYPE HTML>*

*<html>*

*<head>*

*<title>Пример 2\_2</title>*

*</head>*

*<body>*

*<script>*

*var year = prompt('В каком году появилась спецификация ECMA-262 5.1?', '');*

*if (year < 2011) {*

*alert('Это слишком рано..');*

*}*

*else if (year > 2011) {*

*alert('Это поздновато..');*

*}*

*else {*

*alert('Да, точно в этом году!');*

*}*

*</script>*

*</body>*

*</html>*

**Оператор вопросительный знак '?'**

Иногда нужно в зависимости от условия присвоить переменную. Например:

*var access;*

*var age = prompt('Сколько вам лет?', '');*

*if (age > 14) {*

*access = true;*

*} else {*

*access = false;*

*}*

*alert(access);*

Оператор вопросительный знак '?' позволяет делать это короче и проще. Он состоит из трех частей: условие ? значение1 : значение2

Проверяется условие, затем если оно верно — возвращается значение1 , если неверно — значение2, например:

*access = (age > 14) ? true : false;*

Оператор '?' выполняется позже большинства других, в частности — позже сравнений, поэтому скобки можно не ставить:

*access = age > 14 ? true : false;*

Пример кода в составе страницы HTML:

*<!DOCTYPE HTML>*

*<html>*

*<head>*

*<title>Пример 2\_4</title>*

*</head>*

*<body>*

*<script>*

*var access;*

*var access2;*

*var age = prompt('Сколько вам лет?', '');*

*if (age > 14) {*

*access = true;*

*}*

*else*

*{*

*access = false;*

*}*

*access2 = (age > 14) ? true : false;*

*alert("access = "+access+" ; access2 = "+ access2);*

*</script>*

*</body>*

*</html>*

**Несколько операторов '?'**

Несколько операторов if..else можно заменить последовательностью операторов '?'. Например:

*var a = prompt('a?', 1);*

*var res = (a == 1) ? 'значение1' :*

*(a == 2) ? 'значение2' :*

*(a > 2) ? 'значение3' :*

*'значение4';*

*alert(res);*

Вопросительный знак проверяет сначала a == 1, если верно — возвращает значение1, если нет — идет проверять a == 2. Если это верно — возвращает значение2, иначе проверка a > 2 и значение3.. Наконец, если ничего не верно, то значение4. Альтернативный вариант с if..else:

*var res;*

*if (a == 1) {*

*res = 'значение1';*

*} else if (a == 2) {*

*res = 'значение2';*

*} else if (a > 2) {*

*res = 'значение3';*

*} else {*

*res = 'значение4';*

*}*

**Логические операторы**

В JavaScript поддерживаются операторы || (ИЛИ), && (И) и ! (НЕ).

Они называются «логическими», но в JavaScript могут применяться к значениям любого типа и возвращают также значения любого типа.

|| (ИЛИ)

Оператор ИЛИ выглядит как двойной символ вертикальной черты:

*result = a || b;*

Логическое ИЛИ в классическом программировании работает следующим образом: «если хотя бы один из аргументов true, то возвращает true, иначе — false».

Получается следующая таблица результатов:

*alert( true || true ); // true*

*alert( false || true ); // true*

*alert( true || false); // true*

*alert( false || false); // false*

При вычислении ИЛИ в JavaScript можно использовать любые значения. В этом случае они будут интерпретироваться как логические.

Например, число 1 будет воспринято как true, а 0 — как false:

*if ( 1 || 0 ) { // сработает как if( true || false )*

*alert('верно');*

*}*

Обычно оператор ИЛИ используется в if, чтобы проверить, выполняется ли хотя бы одно из условий, например:

*var hour = 9;*

*if (hour < 10 || hour > 18) {*

*alert('Офис до 10 или после 18 закрыт');*

*}*

*Можно передать и больше условий:*

*var hour = 12, isWeekend = true;*

*if (hour < 10 || hour > 18 || isWeekend) {*

*alert('Офис до 10 или после 18 или в выходной закрыт');*

*}*

Пример кода в составе страницы HTML:

*<!DOCTYPE HTML>*

*<html>*

*<head>*

*<title>Пример 2\_5</title>*

*</head>*

*<body>*

*<script>*

*var hour = prompt('Введите интересующий час работы');*

*var isWeekend = prompt('Сегодня выходной? (да – true; нет - false)');*

*if (hour < 10 || hour > 18 || isWeekend) {*

*alert('Офис до 10 или после 18 или в выходной закрыт');*

*}*

*</script>*

*</body>*

*</html>*

**Короткий цикл вычислений**

JavaScript вычисляет несколько ИЛИ слева направо. При этом, чтобы экономить ресурсы, используется так называемый «короткий цикл вычисления». Допустим, вычисляются несколько ИЛИ подряд: a || b || c || .... Если первый аргумент — true, то результат заведомо будет true (хотя бы одно из значений — true), и остальные значения игнорируются.

Это особенно заметно, когда выражение, переданное в качестве второго аргумента, имеет сторонний эффект — например, присваивает переменную. При запуске примера ниже присвоение x не произойдёт:

*var x;*

*true || (x = 1); // просто вычислим ИЛИ, без if*

*alert(x); // undefined, x не присвоен*

А в примере ниже первый аргумент — false, так что ИЛИ попытается вычислить второй, запустив тем самым присваивание:

*var x;*

*false || (x = 1);*

*alert(x); // 1*

***&& (И)***

Оператор И пишется как два амперсанда &&:

*result = a && b;*

В классическом программировании И возвращает true, если оба аргумента истинны, а иначе — false

*alert( true && true ); // true*

*alert( false && true ); // false*

*alert( true && false); // false*

*alert( false && false); // false*

Пример:

*var hour = 12, minute = 30;*

*if (hour == 12 && minute == 30) {*

*alert('Время 12:30');*

*}*

К И применим тот же принцип «короткого цикла вычислений», но немного по-другому, чем к ИЛИ. Если левый аргумент — false, оператор И возвращает его и заканчивает вычисления. Иначе — вычисляет и возвращает правый аргумент.

*// Первый аргумент - true,*

*// Поэтому возвращается второй аргумент*

*alert(1 && 0); // 0*

*alert(1 && 5); // 5*

*// Первый аргумент - false,*

*// Он и возвращается, а второй аргумент игнорируется*

*alert(null && 5); // null*

*alert(0 && "не важно"); // 0*

!!! Приоритет оператора И && больше, чем ИЛИ ||, т.е. он выполняется раньше. Поэтому в следующем коде сначала будет вычислено правое И: 1 && 0 = 0, а уже потом — ИЛИ.

***! (НЕ)***

Оператор НЕ — самый простой. Он получает один аргумент. Синтаксис:

*var result = !value;*

Действия !:

1. Сначала приводит аргумент к логическому типу true/false.

2. Затем возвращает противоположное значение.

Например:

*alert( !true ) // false*

*alert( !0 ) // true*

В частности, двойное НЕ используются для преобразования значений к логическому типу:

*alert( !!"строка" ) // true*

*alert( !!null ) // false*

Объекты различных типов и доступные с ними методы и свойства

В примере ниже показаны виды переменных, которые можно объявить как объекты определенного типа. Так как по сути они будут экземплярами, то они наследуют соответствующий набор методов и свойств, к которым можно обращаться и которые можно использовать.

*<!DOCTYPE HTML>*

*<html>*

*<head> </head>*

*<body>*

*<script>*

*var i = 1;*

*var j;*

*var person = {*

*firstname: "John",*

*lastname: "Doe",*

*id: 5566*

*};*

*var carname = new String;*

*var x = new Number;*

*var y = new Boolean;*

*var cars = new Array;*

*var anything = new Object;*

*</script>*

*</body>*

*</html>*

**Цикл while**

Цикл while имеет вид:

*while (условие) {*

*// код, тело цикла*

*}*

Пока условие верно — выполняется код из тела цикла.

Например, цикл ниже выводит i пока i < 3:

*var i = 0;*

*while (i < 3) {*

*alert(i);*

*i++;*

*}*

Если бы i++ в коде выше не было, то цикл выполнялся бы (в теории) вечно. На практике, браузер выведет сообщение о «зависшем» скрипте и посетитель его остановит.

Пример бесконечного цикла:

*while (true) {*

*// ...*

*}*

Условие в скобках интерпретируется как логическое значение, поэтому вместо while (i!=0) обычно пишут while (i):

*var i = 3;*

*while (i) { // при i=0 значение в скобках будет false и цикл остановится*

*alert(i);*

*i--;*

*}*

*<!DOCTYPE HTML>*

*<html>*

*<head> </head>*

*<body>*

*<script>*

*var i = 1;*

*//Обратите внимание на создание объектного типа и способ обращения к нему в скрипте*

*var phone\_numbers = {*

*firstNumber: "1",*

*twoNumber: "2",*

*threeNumber: "3"*

*};*

*while (i < 10) {*

*//обратите внимание – можно явно преобразовывать число к строке*

*phone\_numbers.firstNumber = phone\_numbers.firstNumber + "-" + i.toString();*

*//но также работает и неявное преобразование*

*phone\_numbers.twoNumber = phone\_numbers.twoNumber + "-" + i;*

*i++;*

*}*

*alert(phone\_numbers.firstNumber);*

*alert(phone\_numbers.twoNumber);*

*alert(i);*

*</script>*

*</body>*

*</html>*

**Цикл do..while**

Проверку условия можно поставить под телом цикла, используя специальный синтаксис

*do..while:*

*do {*

*// тело цикла*

*} while (условие);*

Цикл, описанный, таким образом, сначала выполняет тело, а затем проверяет условие.

Например:

*var i = 0;*

*do {*

*alert(i);*

*i++;*

*} while (i < 3);*

Синтаксис do..while редко используется, т.к. обычный while нагляднее — в нём не приходится искать глазами условие и ломать голову, почему оно проверяется именно в конце.

**Цикл for**

Чаще всего применяется цикл for. Выглядит он так:

*for (начало; условие; шаг) {*

*// ... тело цикла ...*

*}*

Например, цикл ниже выводит значения от 0 до 3 (не включая 3):

*var i;*

*for (i=0; i<3; i++) {*

*alert(i);*

*}*

 Начало i=0 выполняется при заходе в цикл.

 Условие i<3 проверяется перед каждой итерацией.

 Шаг i++ выполняется после каждой итерации, но перед проверкой условия.

В цикле также можно определить переменную:

*for (var i=0; i<3; i++) {*

*//...*

*}*

Любая часть for может быть пропущена.

Например, можно убрать начало:

*var i = 0;*

*for (; i<3; i++)*

*{ //...*

*}*

Можно убрать и шаг:

*var i = 0;*

*for (; i<3; ) {*

*// цикл превратился в аналог while (i<3)*

*}*

А можно и вообще убрать все, получив бесконечный цикл:

*for (;;) {*

*// будет бесконечный цикл*

*}*

**Директивы break и continue**

Для более гибкого управления циклом используются директивы break и continue.

Выход: break

Выйти из цикла можно не только при проверке условия но и, вообще, в любой момент.

Эту возможность обеспечивает директива break.

Например, бесконечный цикл в примере прекратит выполнение при i==5:

*var i=0;*

*while(1) {*

*i++;*

*if (i==5) break;*

*alert(i);*

*}*

*alert('Последняя i = '+ i ); // 5 (\*)*

Выполнение продолжится со строки (\*), следующей за циклом.

Следующая итерация: continue

Директива continue прекращает выполнение текущей итерации цикла. Например, цикл ниже не выводит четные значения:

*for (var i = 0; i < 10; i++) {*

*if (i % 2 == 0) continue;*

*alert(i);*

*}*

Для четных i срабатывает continue, выполнение блока прекращается и управление передается на for. Нельзя использовать break/continue справа от оператора ‘?’ Обычно мы можем заменить if на оператор вопросительный знак '?'. То есть, запись:

*if (условие) {*

*a();*

*} else {*

*b();*

*}*

..Аналогична записи:

*условие ? a() : b();*

В обоих случаях в зависимости от условия выполняется либо a() либо b(). Но разница состоит в том, что оператор вопросительный знак '?', использованный во второй записи, возвращает значение. Синтаксические конструкции, которые не возвращают значений, нельзя использовать в операторе '?'. К таким относятся большинство конструкций и, в частности, break/continue. Поэтому такой код приведёт к ошибке:

*(i > 5) ? alert(i) : continue;*

**Метки**

Бывает нужно выйти одновременно из нескольких уровней цикла.

Например, надо ввести значения в точках с координатами, вот так:

*for (var i = 0; i < 3; i++) {*

*for (var j = 0; j < 3; j++) {*

*var input = prompt('Значение в координатах '+i+','+j, '');*

*if (input == null) break; // (\*)*

*}*

*}*

*alert('Готово!');*

Обычный вызов break в строке (\*) не может прервать два цикла сразу. Для этого существуют метки. Метка имеет вид "имя:", имя должно быть уникальным. Она ставится перед циклом, вот так:

*outer: for (var i = 0; i < 3; i++) { ... }*

Можно также выносить ее на отдельную строку. Вызов break outer прерывает управление цикла с такой меткой, вот так:

*outer:*

*for (var i = 0; i < 3; i++) {*

*for (var j = 0; j < 3; j++) {*

*var input = prompt('Значение в координатах '+i+','+j, '');*

*if (input == null) break outer; // (\*)*

*}*

*}*

*alert('Готово!');*

Директива continue также может быть использована с меткой. Управление перепрыгнет на следующую итерацию цикла с меткой. Метки можно ставить в том числе на блок, без цикла:

*my: {*

*for (;;) {*

*for (i=0; i<10; i++) {*

*if (i>4) break my;*

*}*

*}*

*some\_code; // произвольный участок кода*

*}*

*alert("После my"); // (\*)*

В примере выше, break перепрыгнет через some\_code, выполнение продолжится сразу после блока my, со строки (\*). Возможность ставить метку на блоке используется редко. Обычно метки ставятся перед циклом.

**Конструкция switch**

Конструкция switch заменяет собой сразу несколько if.

Это — более наглядный способ сравнить выражение сразу с несколькими вариантами.

Выглядит она так:

*switch(x) {*

*case 'value1': // if (x === 'value1')*

*...*

*[break]*

*case 'value2': // if (x === 'value2')*

*...*

*[break]*

*default:*

*...*

*[break]*

*}*

 Переменная x проверяется на строгое равенство первому значению value1, затем второму value2 и так далее.

 Если соответствие установлено — switch начинает выполняться от соответствующей директивы case и далее, до ближайшего break (или до конца switch). При этом case называют вариантами switch.

 Если ни один case не совпал — выполняетcя (если есть) вариант default.

Пример использования switch (сработавший код выделен):

*<!DOCTYPE HTML>*

*<html>*

*<head> </head>*

*<body>*

*<script>*

*var a = 2 + 2;*

*switch (a) {*

*case 3:*

*alert('Маловато');*

*break;*

*case 4:*

*alert('В точку!');*

*break;*

*case 5:*

*alert('Перебор');*

*break;*

*default:*

*alert('Я таких значений не знаю');*

*}*

*</script>*

*</body>*

*</html>*

Будет выведено только одно значение, соответствующее 4. После чего break прервёт выполнение.

Если его не прервать — оно пойдёт далее, при этом остальные проверки игнорируются. Например:

*var a = 2+2;*

*switch (a) {*

*case 3:*

*alert('Маловато');*

*case 4:*

*alert('В точку!');*

*case 5:*

*alert('Перебор');*

*default:*

*alert('Я таких значений не знаю');*

*}*

В примере выше последовательно выполнятся три alert.

*alert('В точку!');*

*alert('Перебор');*

*alert('Я таких значений не знаю');*

*В case могут быть любые выражения, в том числе включающие в себя переменные и функции.*

*var a = 1;*

*var b = 0;*

*switch(a) {*

*case b+1:*

*alert(1);*

*break;*

*default:*

*alert('нет-нет, выполнится вариант выше')*

*}*

**Группировка: case**

Несколько значений case можно группировать. В примере ниже case 3 и case 5 выполняют один и тот же код:

*var a = 2+2;*

*switch (a) {*

*case 4:*

*alert('Верно!');*

*break;*

*case 3: // (\*)*

*case 5: // (\*\*)*

*alert('Неверно!');*

*break;*

*default:*

*alert('Я таких значений не знаю');*

*}*

При case 3 выполнение идёт со строки (3) и идёт вниз до ближайшего break, таким образом проходя и то, что предназначено для case 5.

*switch (browser) {*

*case 'IE':*

*alert('О, да у вас IE!');*

*break;*

*case 'Chrome':*

*case 'Firefox':*

*case 'Safari':*

*case 'Opera':*

*alert('Да, и эти браузеры мы поддерживаем');*

*break;*

*default:*

*alert('Мы надеемся, что и в вашем браузере все ок!');*

*}*

**Функции**

Объявление функции

Пример объявления функции:

*function showMessage()*

*{*

*alert('Привет всем присутствующим!');*

*}*

Вначале идет ключевое слово function, после него имя функции, затем список параметров в скобках (в примере выше он пустой) и тело функции — код, который вызывается при её вызове. Объявленная функция доступна по имени, например:

*<!DOCTYPE HTML>*

*<html>*

*<head>*

*<script>*

*function showMessage() {*

*alert('Привет всем присутствующим!');*

*}*

*</script>*

*</head>*

*<body>*

*<script>*

*showMessage();*

*showMessage();*

*</script>*

*</body>*

*</html>*

Внешние переменные

Функция может обратиться ко внешней переменной, например:

*var userName = 'Вася';*

*function showMessage() {*

*var message = 'Привет, я ' + userName;*

*alert(message);*

*}*

*showMessage(); // Привет, я Вася*

Доступ возможен не только на чтение, но и на запись. При этом, так как переменная внешняя, то изменения будут видны и снаружи функции:

*var userName = 'Вася';*

*function showMessage() {*

*userName = 'Петя'; // (1) присвоение во внешнюю переменную*

*var message = 'Привет, я ' + userName;*

*alert(message);*

*}*

*showMessage();*

*alert(userName); // Петя, значение внешней переменной изменено функцией*

Конечно, если бы внутри функции, в строке (1), была бы объявлена своя локальная переменная var userName, то все обращения использовали бы её, и внешняя переменная осталась бы неизменной. Внимание: неявное объявление глобальных переменных! В старом стандарте JavaScript существовала возможность неявного объявления переменных присвоением значения.

Например:

*function showMessage() {*

*message = 'Привет'; // без var!*

*}*

*showMessage();*

*alert(message); // Привет*

В коде выше переменная message нигде не объявлена, а сразу присваивается. Скорее всего, программист просто забыл поставить var.

В современном стандарте JavaScript такое присвоение запрещено, а в старом, который работает в браузерах по умолчанию, переменная будет создана автоматически, причём в примере выше она создаётся не в функции, а на уровне всего скрипта.

**Параметры**

При вызове функции ей можно передать данные, которые та использует по своему усмотрению. Например, этот код выводит два сообщения:

*function showMessage(from, text) { // параметры from, text*

*from = "\*\* " + from + " \*\*"; // здесь может быть сложный код оформления*

*alert(from + '\n\n' + text);*

*}*

*showMessage('Маша', 'Привет!');*

*showMessage('Маша', 'Как дела?');*

Параметры копируются в локальные переменные функции.

В примере ниже изменение from в строке (1) не отразится на значении внешней переменной from (2), т.к. изменена была копия значения:

*function showMessage(from, text) {*

*from = '\*\*' + from + '\*\*'; // (1), красиво оформили from*

*alert(from + '\n\n' + text);*

*}*

*var from = 'Маша', msg = 'Привет!'; // (2)*

*showMessage(from, msg); // значения будут скопированы в параметры*

*alert(from); // перезапись в строке (1) не повлияет на внешнюю переменную*

**Аргументы по умолчанию**

Функцию можно вызвать с любым количеством аргументов.

Например, функцию показа сообщения showMessage(from, text) можно вызвать с одним аргументом:

*showMessage("Маша");*

Если параметр не передан при вызове — он считается равным undefined. Такую ситуацию можно отловить и назначить значение «по умолчанию»:

*<!DOCTYPE HTML>*

*<html>*

*<head> </head>*

*<body>*

*<script>*

*function showMessage(from, text) {*

*if (text == undefined) {*

*text = 'текст не передан';*

*}*

*alert(from + ": " + text);*

*}*

*showMessage("Маша", "Привет!"); // Маша: Привет!*

*showMessage("Маша"); // Маша: текст не передан*

*</script>*

*</body>*

*</html>*

**Возврат значения**

Функция может возвратить результат, который будет передан в вызвавший её код.

Например, создадим функцию calcD, которая будет возвращать дискриминант квадратного уравнения по формуле b

2 - 4ac:

*<!DOCTYPE HTML>*

*<html>*

*<head> </head>*

*<body>*

*<script>*

*function calcD(a, b, c) {*

*return b \* b - 4 \* a \* c;*

*}*

*var test = calcD(-4, 2, 1);*

*alert(test); // 20*

*</script>*

*</body>*

*</html*