**Практическое занятие №10 Управляющие конструкции, массивы и циклы**

**Задание №1**

Убедитесь в том что у Вас запущен Open Server и папка с материалами урока размещена в директории /domains сервера. Заходим на <http://js10.loc> и начинаем урок.

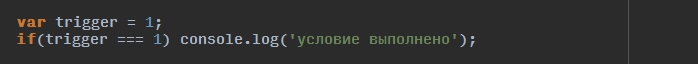
**Задание №2**

Часто возникает необходимость выполнить какие-то действия только при соблюдении одного или нескольких условий. «Проверка условия» представляет собой операцию, в результате которой возвращается значение логического типа Boolean. Это может любая проверка – наличие переменной или ее значения, сравнение двух переменных и т.д.

Ключевое слово if принимает некое выражение для проверки, и если выражение возвращает true, выполняется блок инструкций, стоящий за этим ключевым словом.

Совместно с ключевым словом else предыдущая конструкция работает как триггер (переключатель). Если выражение проверки вернуло false, то выполняется блок инструкций, стоящий после else . Выполниться может всегда только один блок инструкций в зависимости от результатов проверки условия.

Создадим простой блок if:

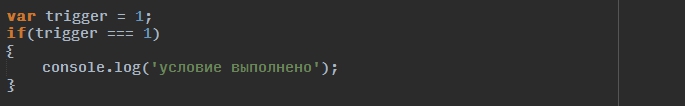


Как Вы видите инструкция console.log выполняется. Потому что условие цикла было true. Это легко проверить, выполнив следующую инструкцию перед if:

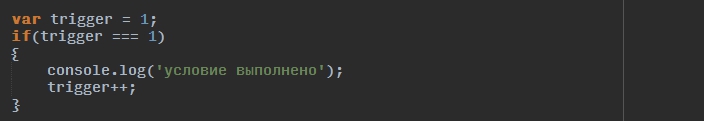
http://dl4.joxi.net/drive/2017/05/29/0004/3207/285831/31/7f125e2ed6.jpg

В текущем варианте инструкция с if включает в себя только одну инструкцию сразу после себя. Что бы разместить в if несколько инструкций нужно создать блок инструкций. Для создания блока инструкций применяют фигурные скобки. Такое объединение необходимо в составных инструкциях и функциях, которые будут рассмотрены в этом уроке. Нужно понимать, что блок инструкций будет выполняться лишь тогда, когда программа получит указание выполнить его.

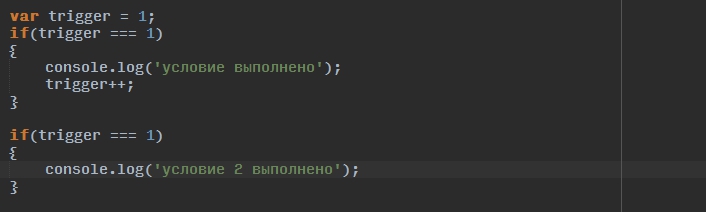
Добавим блок инструкций для нашего if:



В принципе ничего не поменялось, но мы получили возможность добавления других инструкций в if, сделаем это:

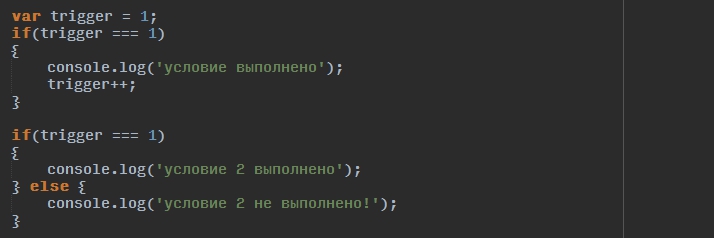


Теперь в блоке if будут выполняться две инструкции – одна на вывод в консоль, другая на увеличение переменной «trigger». Что бы это проверить создадим идентичный блок if ниже:



Как Вы видите условие 2 уже не выполняется, потому что переменная была увеличена в первом блоке if.

Ко второму блоку if добавляем блок else, который срабатывает если условие If не выполнено (false).



**Задание №3**

Даны два числа number1 и number2, используя конструкцию if else выведите в консоль большее из них.

**Задание №4**

Даны две переменные min и max. С помощью if else перераспределите значения этих переменных так, что бы в min было меньшее значение, а в max – большее.

**Задание №5**

Дано целое число в переменной number, используя if else вывести к консоль его описание: «отрицательное», «нулевое», «положительное».

**Задание №6**

Даны два числа в переменных number1 и number2, с помощью if else проверить если их значения не равны, то каждой переменной присвоить большее из значений.

**Задание №7**

Даны три числа (number1, number2, number3), используя конструкцию if else найти и вывести наибольшее из них.

**Задание №8**

Даны три числа (number1, number2, number3), используя конструкцию if else найти сумму двух наибольших из них.

**Задание №9**

Дано целое число number, с помощью конструкции if else вывести его строку описание: «четное/нечетное», «однозначное, двухзначное, трёхзначное».

**Задание №10**

Условный (тернарный) оператор - единственный оператор в JavaScript, принимающий три операнда. Он часто используется в качестве укороченного варианта условного оператора if.

Оператор возвращает значение *выражения1*, если условие верно, и значение *выражения2* в противном случае.

http://dl4.joxi.net/drive/2017/05/29/0004/3207/285831/31/ab2d4cf7af.jpg

Протестируйте пример кода выше и выведите значение переменной в консоль, так же измените условие на false и протестируйте как измениться значение переменной result.

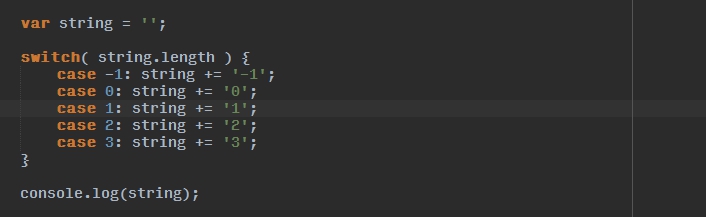
**Задание №11 и №12**

Замените конструкции if else, которая уже присутствует в файле задания на тернарные операторы, так что бы результат работы остался прежним.

**Задание №13**

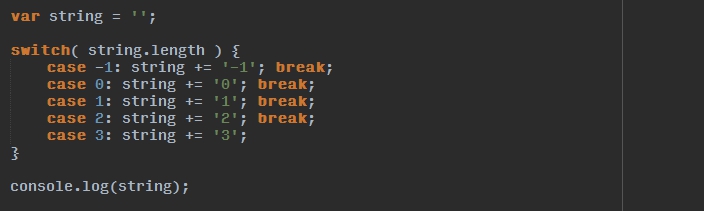
Часто применяется конструкция переключения SWITCH – выбор между заранее заданными вариантами. Эта инструкция может быть полезна в случае заранее ограниченного выбора. Синтаксис этой инструкции отличается от других и немного более сложен (наборы инструкций не объединяются в блоки).

В этом задании мы создадим и разберём особенности конструкции SWITCH. Создаём каркас:



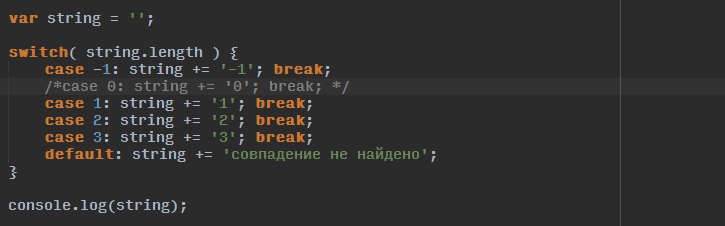
Разбираем получившийся результат примера. Очевидно что выражение «string.lenght» вернёт 0, далее конструкция SWITCH подхватывает это значение и начинает его сравнивать (==) со значениями стоящими после «case» последовательно сверху вниз. Если найдено совпадение для примера это «case 0:», то будет выполнен код за двоеточием и ВЕСЬ КОД НИЖЕ! (первое попадание и всё что за ним).

Если нам не нужно выполнение последующих операторов «case» нам нужно добавить в конце каждого инструкцию «break», как показано ниже:



Отследите как поменялось поведение конструкции SWITCH.

Инструкция default нужно для того, чтобы выполнился код в том случае, когда не найдено ни одно совпадение. Например, вывел бы отрицательный результат поиска. Применение default необязательно. Это ключевое слово может стоять в любом месте, а не только после всех case (в этом случае следует заканчивать стоящий после него код инструкцией break ). Добавим инструкцию default к нашему примеру:



**Задание №14**

Дано целое число day от 1 до 7, используя SWITCH выведите название дня недели по значению day, если day > 7, то выведите сообщение что «день недели не определён». Самостоятельно протестируйте свой функционал в различных значениях day.

**Задание №15**

Дан номер месяца, с помощью SWITCH выведите название соответствующего времени года. Если month > 12, то выведите сообщение что «месяц не определён». Самостоятельно протестируйте свой функционал в различных значениях month.

**Задание №16**

В переменной $value находится вес объекта, в переменной $format – идентификатор системы измерения веса (1 – кг, 2 – граммы, 3 – миллиграммы, 4 – центнер, 5 - тонны). С помощью SWITCH создайте скрипт, который будет выводить скрипт в консоль браузера только в килограммах, независимо от того какой начальный формат веса был передан. Самостоятельно протестируйте свой функционал в различных значениях $value и $format.

**Задание №17**

В переменной $count дается количество товаров в корзине. С помощью SWITCH выведите правильное склонение слова «товар» в зависимости от числа $count (1 – товар, 2, 3, 4 – товара, все что больше - товаров). Самостоятельно протестируйте свой функционал в различных значениях $ count.

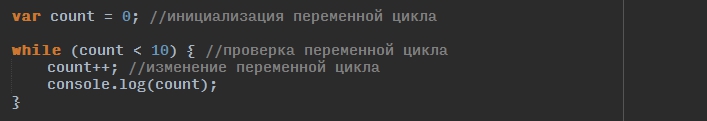
**Задание №18**

Для выполнения повторяющихся действий применяются инструкции цикла. Любая такая инструкция связана, прежде всего, с тремя обязательными шагами:

* Инициализация переменной (или переменной цикла)
* Проверка условия, связанного с переменной цикла
* Изменение переменной цикла

Любая инструкция цикла выполняет блок инструкций, если проверка условия возвращает true. В противном случае выполнение цикла прекращается.

Создадим простейший цикл WHILE:



Проверьте работоспособность своего кода! Попробуйте самостоятельно изменить условие цикла, так что бы он ни разу не выполнился.

Главное отличие цикла do while от цикла while – выполнение блока инструкций хотя бы один раз. Проверка условия стоит после исполняемого кода. Такая задача очень специфична. Небольшие отличия – “;” в конце этой инструкции, поскольку она заканчивается проверкой, а блок инструкций стоит до проверки. Создадим простейший цикл DO WHILE:



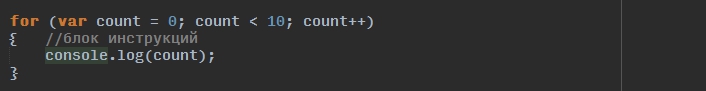
**Задание №19**

Дана начальная сумма вклада $summa, Известно, что каждый месяц ТЕКУЩАЯ сумма вклада увеличивается на заданный процент $persent (от 1 до 100). С помощью цикла WHILE нужно написать скрипт, который будет считать сколько полных месяцев потребуется для того что бы удвоить вклад.

**Задание №20**

Основное отличие цикла FOR от цикла WHILE – инициализация, проверка и изменение переменной цикла стоят в одном месте, что облегчает код. В разделе инициализации можно применить оператор “,”, объединив инициализацию или объявление нескольких переменных. Ровно так же можно поступить и в разделе изменения – с помощью того же оператора “,” можно изменять значения нескольких переменных.

Создадим пример простейшего цикла FOR:



Проверьте работоспособность своего цикла.

**Задание №21**

С помощью цикла FOR выведите в консоль сколько нужно заплатить за 1-10 единиц товара, если известна его стоимость $price и так же известно, что от трёх единиц дается скидка в 10%, а более 5 ед. – скидка 20%, более 7 ед. – 25%.

**Задание №22**

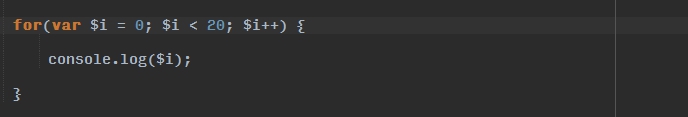
Даны два числа A и B, c помощью цикла FOR найдите сумму всех целых чисел от А до В включительно и выведите в консоль. Учитывайте, что в А и В могут быть абсолютно случайные числа.

**Задание №23**

Инструкции ***break***вызывает прекращение выполнения цикла (все инструкции, идущие за break в теле цикла, игнорируются) и переход к коду, стоящему за этим циклом.

Инструкция ***continue***применяется только в циклах. Встретив ее, программа игнорирует все инструкции, стоящие в теле цикла за continue и переходит к следующей проверке условия цикла.

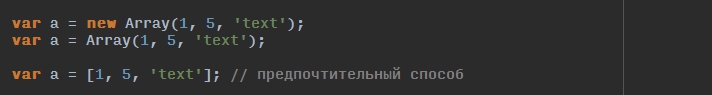
Создадим простой цикл, как показано ниже:



Теперь самостоятельно модифицируйте поведение этого цикла так что бы он выводил только четные значения в консоль (используйте continue) и прервать цикл, когда значение переменной $i будет больше 10 (break).

**Задание №24**

Массив – это коллекция пронумерованных элементов. Создать массив можно несколькими способами:



Если не передать никаких значений в класс-конструктор или ничего не писать внутри квадратных скобок, будет создан пустой массив.

Элементы массива имеют так называемые индексы – его номер по порядку. Нумерация в массиве начинается с нуля. Доступ к элементу массива осуществляется с помощью квадратных скобок и его индекса в массиве.

http://dl3.joxi.net/drive/2017/05/30/0004/3207/285831/31/f338dedb4b.jpg

Создать в массиве новый элемент или переопределить значение уже существующего можно с помощью оператора присваивания.

http://dl4.joxi.net/drive/2017/05/30/0004/3207/285831/31/35841ff315.jpg

Значениями элементов массива могут быть любые типы данных.

**Задание №25**

Каждый экземпляр Array наследует все свойства и методы своего класса-конструктора. В этом задании мы познакомимся со свойством length, которое возвращает индекс последнего элемента массива + 1. Рассмотрим это свойство на практике:



**Задание №26**

Используя цикл, создайте массив со значениями всех нечетных цифр из диапазона от 1 до 101 включительно и выведите его в консоль.