Javascript. Условные операторы. Циклы.

План урока

1. Способы получить данные пользователя
2. Операторы
3. Функции для работы с числами
4. Методы работы со строками
5. Инструкции if / else
6. Инструкции switch / case
7. Тернарный оператор
8. Что такое цикл?
9. Цикл while
10. Цикл do / while
11. Цикл for
12. Инкременты/декременты.
13. Метки break и continue;
14. Практическое применение циклов
15. Распространенные ошибки в циклах

Для получения данных от пользователя мы, пока что, будем использовать `prompt()` и `confirm()`.

- **confirm** - выводит модальное окно с сообщением, и 2 кнопки, ok и cancel. При нажатии на ok возвращается true, при нажатии на cancel возвращается false.

- **prompt** - выводит модальное окно с полем ввода и кнопками ok/cancel. При ok возвращает то что было введено в поле ввода, при cancel возвращает null.

Реализуем простое применение данных функций

```
const acceptation = confirm(‘Пожалуйста подтвердите присутствие!’);
console.log(acceptation);
```

```
const userName = prompt(‘Пожалуйста введите имя:', '');
console.log(userName);
```

Базовые операции

Синтаксис	Операция
a + b	Сложение
a - b	Вычитание
a * b	Умножение
a / b	Деление
a % b	Остаток от деления

Операторы сравнения

– Оператор нестрогого равенства **==**

5 == 5 // true, 10 == ’10’ // true,

– Оператор строгого равенства **===**

’10’ **===** 10 // false

5 === 5 // true

– Чтобы определить неравенство элементов можем использовать соответственно **!=** и **!==**

Операторы ‘>’ ‘<‘ ‘>=‘ ‘<=‘

– Показывает какой элемент больше или меньше

– Может сравнивать строки

– Не ассоциативный -10 < -5 < 0 вернет false

– Сравнение строк происходит по unicode

– Маленькие буквы всегда больше больших

Логические операторы &&, ll и !

Оператор	Описание
&&	логическое "И". Возвращает одно из значений (операндов) - левый операнд если его можно привести к false, и правый в остальных случаях
||	логическое "ИЛИ". Возвращает одно из значений (операндов) - левый операнд если его можно привести к true, и правый в остальных случаях
!	Возвращает false если операнд приводится к true, и true, если операнд приводится к false

Эти значения всегда приводятся к false

- **0**
- **-0**
- **NaN**
- **null**
- **undefined**
- **""** или **''** или **``**
- **false**

Функции для работы с числами

Функция **parseInt()** принимает строку в качестве аргумента и возвращает целое число в соответствии с указанным основанием системы счисления.

Синтаксис

```
parseInt(string, radix);
```

Функция **`parseFloat()`** принимает строку в качестве аргумента и возвращает десятичное число (число с плавающей точкой)

Синтаксис

```
parseFloat(строка)
```

Сложение чисел с плавающей точкой

Есть очень важная "особенность" при сложении не целых чисел в JavaScript. Результат сложения **`0.1`** и **`0.2`** немного больше чем **`0.3`**. В то время как мы считаем в десятичной системе, машина считает в двоичной системе.

```
const result = 0.1 + 0.2;
// result == 0.30000000000000004;
```

Решение

```
const result = (0.1 + 0.2).toFixed(10);
// result == 0.3000000000;
```

Методы для работы со строками

Синтаксис	Функционал
`length`	свойство строки, вернет длину строки
`toLowerCase` и `toUpperCase`	вернут строку в соответствующем регистре
`charAt`	вернет символ который находится на позиции с индексом который указывам при вызове метода
`indexOf`	вернет позицию на которой находится подстрока или -1, если ничего не найдено
`startsWith`	проверяет начинается ли строка на указаную подстроку, возвращает true или false
`endsWith`	проверяет заканчивается ли строка на указаную подстроку, возвращает true или false
`includes`	один из наиболее часто используемых методов, в большинстве случаев заменяет indexOf, проверяет входит ли подстрока в строку, возвращает true или false
`trim`	очень полезный метод, удаляет пробельные символы в начале и конце строки

Конструкция if / else if/ else

```
let a = 5;
// a = 15;
// a = 25;
if(a === 5){
    console.log("Мы получили число 5!");
} else if(a === 15) {
    console.log("Мы получили число 15!");
} else {
    console.log("Мы получили какое-то другое число");
}
```

Условные конструкции switch / case

```
let a = 5;
// a = 15;
// a = 25;

switch(a) {
    case 5: {
        console.log(“Мы получили число 5!”);
        break; // Без этой метки, выполнение кода продолжится!
    }
    case 15: {
        console.log(“Мы получили число 15!”);
        break; // Без этой метки, выполнение кода продолжится!
    }
    default: {
        console.log(“Мы получили какое-то другое число”);
    }
}
```

Тернарный оператор

Сокращенная форма записи условия if/else:

```
const a = 0;
if(a){
    const b = true;
} else {
    const b = false;
}

// Или через тернарный оператор
const b = a ? true : false;
```

Создать программу 1:

1. Пользователь вводит число.
2. Если число четное, мы вернем его разделив на 2
3. Если число нечетное, мы вернем его умножив на 2

Создать программу 2:

1. Пользователь вводит число
2. Нужно вернуть корень из этого числа

Используем функцию Math.sqrt();

Обязательно проверяем получаемые данные.

Создать программу 3:

1. Просит ввести число от 1 до 100;
2. Если число не входит в рамки указанного – программа снова просит ввести число
3. Если второй раз число не было введено верно: программа прекращает играть, словно «обижается»
4. Когда число было введено, с помощью функции Math.random – создать число от 1 до 100.
5. Отобразить результат. Число больше выбранного компьютером, число меньше выбранного компьютером или число равно.

Создать программу 4:

1. Пользователь вводит число от 1 до 12.
2. Если число выходит за рамки – сообщение об ошибке(alert).
3. Каждому числу соответсвует название месяца, его нужно вернуть.

Подсказка: switch/case

Перерыв 5 минут

Что такое цикл?

Цикл – это повторяющиеся действия,
определенное количество раз

Примеры циклов

1. Дни недели
2. Отжимания в спортзале
3. Расписание поездов
4. И т.д

Цикл в Javascript

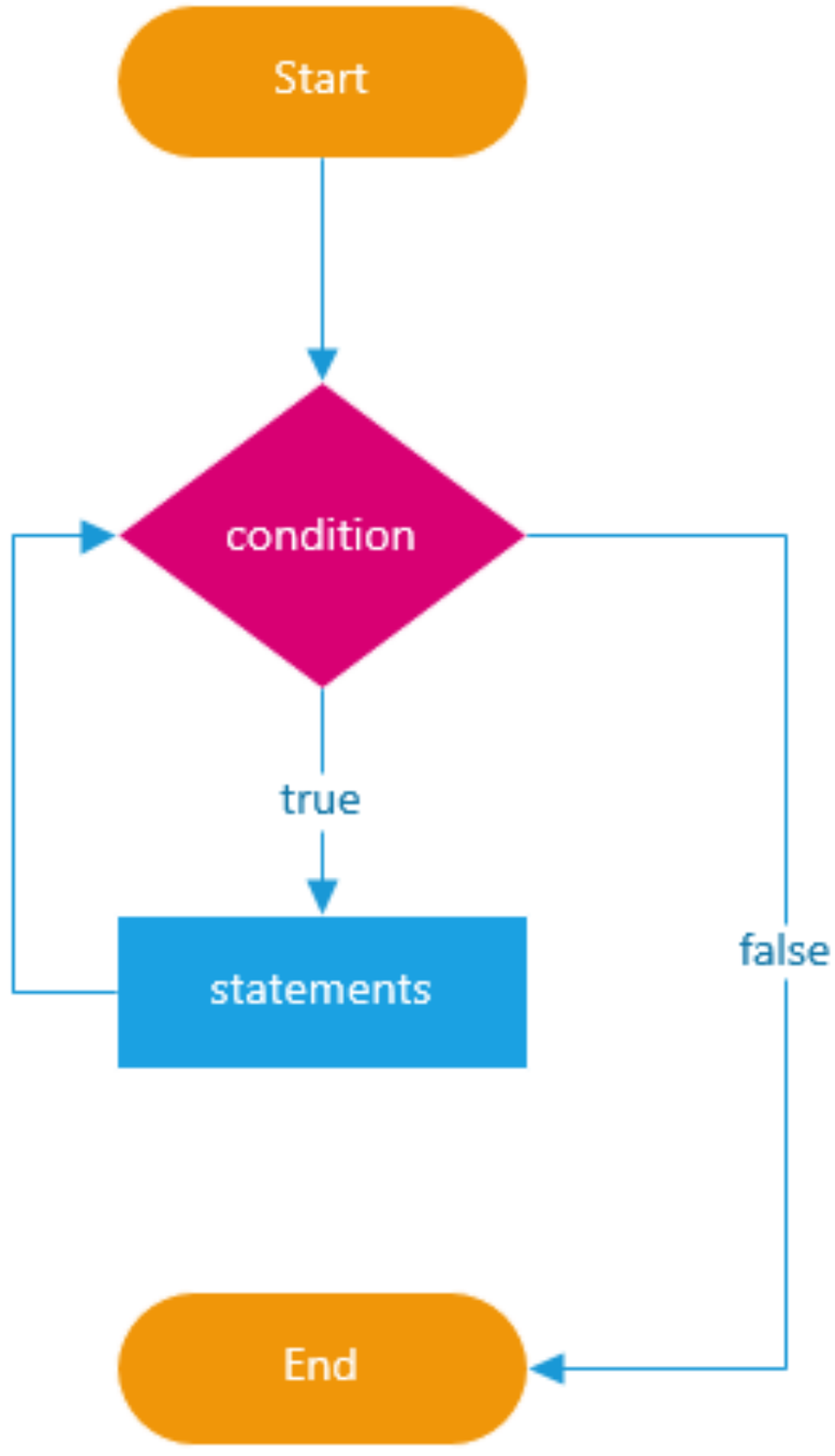
```
let count  = 5;
let a = 1;
if(a < count) { // a == 1
    document.write("a меньше count");
    a = a + 1; // Увеличиваем a на 1
}
...
if(a < count) { // a == 5
    document.write("a меньше count");
    a = a + 1; // Увеличиваем a на 1
}
```

==

```
let count  = 5;
let a = 1;
while(a < count){
    document.write("a меньше count");
    a = a + 1; // Увеличиваем a на 1
}
```

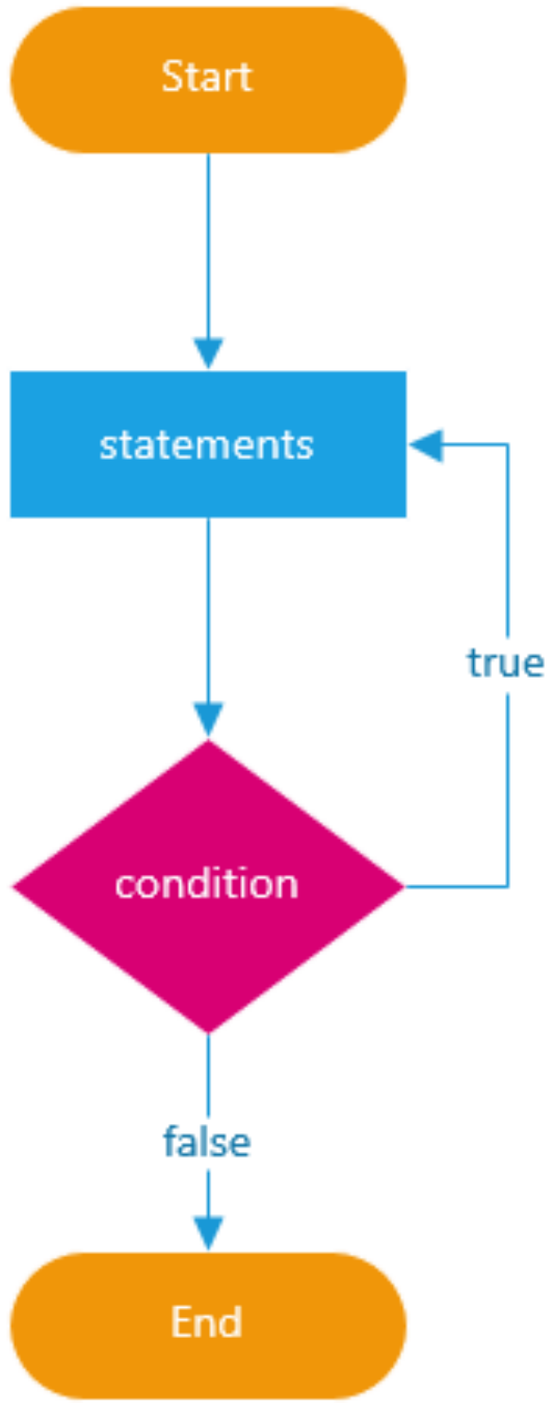
Цикл while

```
while(условие){
      инструкции
}
```



Цикл do / while

```
do {
    инструкции
} while(условие)
```



Пример применения цикла do / while

```
let input = 0;

do {
  input = prompt('Введите число больше 10', '');
} while(input <= 10);
```

Будет выводиться окно с просьбой ввести число,
пока число не будет больше 10

Цикл for

```
for (инициализация; условие; пост-выражение) {
  Инструкции или тело цикла
}
```

- Инициализация (initialization) - выражение инициализации выполняется один раз, когда начинается цикл. Используется для инициализации переменной-счетчика.
- Условие (condition) - представляет собой выражение, оцениваемое перед каждой итерацией цикла.
- Тело (выражение, statements) - выполняется в случае удовлетворения условия.
- Пост-выражение (post-expression) - выполняется после тела на каждой итерации цикла, но перед проверкой условия. Как правило, используется для обновления переменной-счетчика.

Инкременты / декременты

- i++, то же самое что и i += 1 или i = i + 1
- i++ сначала вернет значение, потом выполнит сложение
- ++i сначала выполнит сложение, потом вернет значение

Напишем цикл вместе

Давайте посчитаем сумму значений от 1 до 100.

С чего начнем?

Метки break / continue

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

Директива **`break`** полностью прекращает выполнение цикла и передаёт управление на строку за его телом

Директива **`continue`** прекращает выполнение текущей итерации цикла

Примеры break / continue

```
for(var i = 0; i < 10; i++){
    // Прервемся на 5 итерации цикла.
    if(i == 5) break;
    document.write("Число: " + i);
}
```

```
for(var i = 0; i < 10; i++){
    // Пропустим 5 итерацию цикла.
    if(i == 5) continue;
    document.write("Число: " + i);
}
```

Распространенные ошибки

1. Бесконечный цикл
2. «Потеря» переменной в области видимости
3. Попытка создать «долгий» цикл

Задача на циклы

1) Посчитаем сумму нечетных чисел от 1 до 100.
2) Посчитаем сумму чисел от 100 до 1, но если сумма превысит 1000, то считать дальше не будем