ЛАБОРАТОРНАЯ РАБОТА № 3

Циклы и ветвление кода

Инструкции циклов и ветвления кода называют *управляющими инструкциями*, поскольку они позволяют управлять последовательностью выполнения программ JavaScript. *Инструкции ветвления* используются для создания различных путей выполнения программы в зависимости от логики условий.

Циклы — это простейший способ группирования JavaScript-инструкций в программе.

При разработке логики JavaScript-программ часто наступает момент, когда для определения дальнейшего пути выполнения требуется сделать определенный выбор.

`if...else`

Инструкции `if` и `else` работают совместно, вычисляя логическое выражение и выполняя различные инструкции в зависимости от результата. Инструкции `if` часто используются без инструкции `else`, тогда как последняя должна использоваться только совместно с инструкцией `if`.

Вот как выглядит базовый синтаксис инструкции `if`.

```
if (условие) {
}
```

В данном случае условие — это любое выражение, вычисление которого дает булево значение (`true` или `false`). Если результат вычисления выражения равен `true`, то выполняются инструкции, заключенные в фигурные скобки. Если результат равен `false`, то эти инструкции пропускаются.

Инструкция `else` пригодится в тех случаях, когда требуется выполнить какие-либо другие действия, даже если результат вычисления условия равен `false`.

```
var age = 17;
```

```
if (age < 18) {
document.write ("Продажа алкоголя несовершеннолетним запрещена.");
} else {
document.write ("Чем это закончится?");
}
```

Можно несколько раз подряд использовать if и else с пробелом между ними.

```
let age = prompt('Сколько тебе лет?')
if (age > 18) {
    alert('Мамочки! Ты староват!')
}
else if (age < 18) {
    alert('Эх, мелочь, а приятно... ')
}
else if (age == 18) {
    alert('Самое то, наслаждайся моментом!')
}
```

Сокращенная запись инструкции if...else

Условный оператор `if`, можно записать через *тернарный оператор - Условие ? если условие true : если условие false*

```
prompt("Как тебя зовут?", "Введи, пожалуйста, сюда своё имя") ? document.write("Привет, ", Uname) : alert("Не хочешь представляться??? Ну и не надо! Пока!");
```

Инструкция switch

Инструкция `switch` (переключатель) позволяет выбрать, какая из нескольких инструкций должна быть выполнена, исходя из возможных

значений заданного выражения. Каждое из таких значений называется *вариантом* (case).

Инструкция `switch` имеет следующий синтаксис.

```
switch (выражение) {
case значение 1:
// Инструкции
break;
case значение 2:
// Инструкции
break;
case значение 3:
// Инструкции
break;
default:
// Инструкции
break;
}
```

Обратите внимание на инструкцию `break`, которая указывается вслед за каждой группой инструкций, соответствующей определенному варианту. Каждая такая инструкция `break` обеспечивает прекращение выполнения инструкции `switch` и выход за ее пределы. В случае отсутствия инструкции `break` инструкция `switch` продолжит свою работу и выполнит операторы следующего варианта, независимо от того, совпадает ли указанное в нем значение со значением выражения, указанного в начале инструкции `switch`.

Если ни для одного из предложений `case` совпадений со значением выражения не найдено, то инструкция `switch` осуществляет поиск варианта `default` и выполняет содержащиеся в нем инструкции.

Единственный случай, когда инструкция `break` может быть безболезненно опущена, относится к варианту `default`, указанному

последним среди всех остальных (как и должно быть), поскольку вслед за инструкцией switch выполнение программы продолжается в любом случае.

```
switch (true) {
    case (age > 18):
        alert('Мамочки! Ты староват!')
    break;
    case (age < 18):
        alert('Эх, мелочь, а приятно... ')
    break;
    default:
        alert('Самое то, наслаждайся моментом!')
}
```

```
let languagePreference = prompt('Какой язык планируешь изучить?');
    switch (languagePreference) {
        case 'английский':
            console. log ('Hello!');
        break;
        case 'испанский':
            console. log ('Hola!');
        break;
        case 'немецкий':
            console. log ('Guten Tag!');
        break;
        case 'французский':
            console. log ('Bon Jour!');
        break;
        default:
```

```
            console. log ('Извините, я не знаю ' +
languagePreference + ' язык');
        }
```

Циклы предназначены для многократного выполнения одного и того же набора инструкций. В JavaScript возможны циклы нескольких видов:

- `for`
- `for … in`
- `do...while`
- `while`

Инструкция `for` создает цикл с использованием трех выражений.

- *Инициализация*. Инициализация переменной, обычно — счетчика.
- *Условие*. Булево выражение, которое вычисляется на каждой итерации цикла.
- *Завершающее выражение.* Выражение, которое вычисляется по завершении каждой итерации цикла.

Несмотря на то, что любое из этих выражений может быть опущено, в инструкцию почти всегда включают все три выражения. Обычно цикл `for` используется для выполнения кода заданное количество раз.

Ниже приведен пример использования цикла `for`.

```
for (var x = 1; x < 10; x++){
    console. log (x);
}
```

Рассмотрим выполнение этого цикла более подробно.

1. Переменная, в данном случае x, инициализируется значением 1.
2. Осуществляется проверка того, что x меньше 10. Если это так, то выполняются инструкции тела цикла (в данном случае — инструкция `console. log`).

3. Значение x увеличивается на 1 с помощью оператора инкремента (++).

4. Вновь осуществляется проверка того, что x меньше 10. Если это так, то выполняются инструкции тела цикла.

5. Проверка повторяется до тех пор, пока результат вычисления выражения условия не станет ложным.

```
for (var x = 1; x < 10; x+=2){
    console. log (x);
}
```

Циклы можно использовать для перечисления содержимого массива, сравнивая значение счетчика со значением свойства `length` массива.

```
let areaCodes = ['254','287', '322', '378', '441',
'448', '599', '372'];
    for (x = 0; x < areaCodes.length; x++) {
        document. write ('Другой телефонный код: '
+ areaCodes [ x ] + '<br>');
}
```

for...in

Цикл `for...in` перебирает все свойства объекта. Его также можно использовать для перебора всех элементов массива.

Цикл `for...in` имеет одну интересную особенность. Он совершенно не учитывает, в каком порядке хранятся свойства объекта или элементы массива.

```
let obj = {
    'name' : 'Richard',
    'age' : 41,
    'number' : 83452649871
    }
    for(let key in obj) {
```

```
            document.write(key + '<br>');
        }
    for(let key in obj) {
            document.write(obj[key] + '<br>');
        }
    for(let key in obj) {
            document.write(`Ключ  ${key},  значение
${obj[key]}` + '<br>');//используем обратные кавычки
            }
```

for...of

Цикл, который помогает при работе с массивами.

```
    let areaAge = ['25','37', '32', '38', '42', '48',
'99', '72'];
        for (let item  of areaAge) {
            document. write (item + '<br>');
    }
```

while

Инструкция while создает цикл, который выполняется до тех пор, пока вычисление выражения, устанавливающего условие выполнения цикла, дает истинное значение.

```
    let guessedWord = prompt ('Какое слово я
задумал?');
        while (guessedWord != 'сэндвич') {//пока не
названо слово "сэндвич"
            prompt ('Нет. Это не оно. Попробуйте еще
раз.');
    }
        alert ('Поздравляю! Это именно то слово!');
//выполняется после выхода из цикла
```

do...while

Цикл **do...while** во многом работает точно так же, как и цикл while, за исключением того, что в данном случае инструкции предшествуют проверке условия. Как следствие, данный вид цикла всегда выполняется по крайней мере один раз.

```
let i = 0;
    do {
        i++;
        document. write (i + '<br>');
    } while (i < 10);

let count = 5;
    do {
        document.write(count + '<br>');
    } while (count > 7)

let count = 10;
    do {
        document.write(count + '<br>');
        count--;
    } while (count > 7)
```

break и continue

Инструкции `break` и `continue` прерывают выполнение цикла. Мы уже рассматривали инструкцию `break` в контексте инструкции `switch`, где она использовалась для продолжения программы после выполнения подходящего варианта.

Инструкция `break` в циклах делает то же самое. Она приводит к немедленному выходу из тела цикла, независимо от того, выполняются или не выполняются условия для продолжения выполнения цикла.

```
let count = 0;
    while (count < 10) {
        document.write(count + '<br>');
        count++;
    }
```

Добавьте команду и сравните результаты:

```
let count = 0;
    while (count < 10) {
    if (count == 6) {
        break;}
        document.write(count + '<br>');
        count++;
    }
```

Когда встречается инструкция continue, выполнение текущей итерации прекращается, инструкции цикла, следующие за инструкцией continue, игнорируются, и начинается выполнение следующей итерации цикла.

```
for (let i = 0; i <= 20; i++) {
    if (i%2 == 0){
        continue;
    }
    document. write (i + '<br>');//выводит только нечетные числа
```

Рассмотрим ещё несколько примеров для получения элементов из массива.

```
let arr = ['28', '15', '25', '81', '98'];
    arr.forEach (function (item, i, array) {
        document.write(`Элемент: ${item} <br>
```

```
        Индекс: ${i} <br>
        Массив: ${array} <br>`)
}

let people = [
    {id: 1, name: 'Alex'},
    {id: 2, name: 'Dimon'},
    {id: 3, name: 'Troy'},
    {id: 4, name: 'Ulya'},
    {id: 5, name: 'Mira'},
    ];
    let a = people.find(function(item){
        if(item.id == 4) return item
    })
console.log(a);

let people = [
    {id: 1, name: 'Alex'},
    {id: 2, name: 'Dimon'},
    {id: 3, name: 'Troy'},
    {id: 4, name: 'Ulya'},
    {id: 5, name: 'Mira'},
    ];
    let a = people.findIndex(function(item){
        if(item.id == 2) return item
    })
console.log(a);

let people = [
    {id: 1, name: 'Alex'},
```

```
    {id: 2, name: 'Dimon'},
    {id: 3, name: 'Troy'},
    {id: 4, name: 'Ulya'},
    {id: 5, name: 'Mira'},
    ];
    let a = people.filter(function(item){
        if(item.id < 3) return item
    })
console.log(a);

let arr = ['28', '15', '25', '81', '98'];
    let newArray = arr.map (function (item) {
        return item*3
    })
console.log(newArray);
```

Рассмотрим несколько задач.

Создать массив с четными числами меньше 20.

```
let arr = [];
    for(let i = 0; i < 20; i+=2){
        arr.push(i)
    }
    console.log(arr);
```

Создать массив с числами от 30 до 0, кратные 3.

```
let arr = [];
    for(let i = 30; i > 0; i--){
        if(i%3 == 0){
    arr.push(i)
}}
console.log(arr);
```

В произвольном массиве подсчитать количество четных чисел.

```
let arr = [2,5,6,8,4,9,12,14,11,13,16,15,18];
    let count = 0;
    for(let i = 0; i < arr.length; i++){
        if(arr[i]%2 == 0){
            count++;
        }}
    alert(count);
```