skillUp

Операторы

План

- условный оператор if
- логические операторы
- оператор switch
- операторы цикла while, for
- функции

Оператор if

- Условные операторы используются для выполнения различных действий на основе различных условий. Если результат вычисления условия (выражение приводится к логическому типу) является true то выполняется выражение1, если false то выражение2.

```
if (условие) {

   выражение1}

[Else {

 выражение2}]
```

В логическом контексте:
Число 0, пустая строка "", null и undefined, а также NaN является false,
Остальные значения - true.

Если есть только выражение1 то можно писать без {}
if (true) x = 1;

if..else

необязательный блок else исполнится если условие false

```
if (hour <18) {
    greeting = "Good day";
} Else {
greeting = "Good evening";

}
```

Несколько условий, else if

- используется если нужно проверить несколько условий

```
if (time < 10) {
    greeting = "Good morning";
} else if (time < 20) {
    greeting = "Good day";
} else {
    greeting = "Good evening";
}
```

Сравнение с тернарным оператором

- запись через if

```
if (age > 18) {
  access = true;
} else {
  access = false;
}
```

- запись через ?

```
access = (age > 18) ? true : false;
```

Практика

- написать оператор if..else который принимает значение с prompt и выводит в консоль

1, если значение больше нуля,
1, если значение меньше нуля,
0, если значение равно нулю.

- переписать в тернарный оператор

```
var a = 1;
var n;
if (a> 0) {n = true; }
else {n = false; }
```

Логические операторы

&&	логическое И
||	логическое ИЛИ
!	логическое НЕТ

https://learn.javascript.ru/logical-ops

Логическое И (&&)

- возвращает true, если оба аргумента истинные, а иначе - false

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

- Если левый аргумент - false, оператор && возвращает его и заканчивает вычисления. Иначе - вычисляет и возвращает прав аргумент.

```
alert( 1 && 0 ); // 0
alert( 1 && 5 ); // 5
```

Логическое ИЛИ (||)

- если хотя бы один из аргументов true, то возвращает true, иначе - false "

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

- оператор ИЛИ вычисляет ровно столько значений, сколько необходимо - до первого true.

```
alert( 1 || 0 ); // 1
alert( true || 'неважно что' ); // true
```

Логическое НЕТ (!)

Сначала приводит аргумент к логическому типу true / false.

Затем возвращает противоположное значение.

```
alert (! true); // false
alert (! 0); // true
```

Двойное ни используется чтобы привести значение к булевого типа

```
alert (!! "non-empty string") // true
alert (!! null); // false
```

Практика

a = 5; b = 3;

(a>b) && (a===b)

true && 0 && (‘a’ < ‘Z’)

(a>b) || true || (‘2’==2) || (false == ‘’)

(a<b) && (0 == false)

!(2==2) || (true && ‘’)

Оператор switch

- заменяет собой несколько проверок if

```
switch(x) {
  case 'value1':  // if (x === 'value1')
    ...
    [break]

  case 'value2':  // if (x === 'value2')
    ...
    [break]

  default:
    ...
    [break]
}
```

https://www.w3schools.com/js/js_switch.asp

```
let 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');
}
```

Как работает switch

- Переменная x проверяется на строгую равенство первому значению value1, потом второго value2 и так далее.
- Если соответствие установлено - switch начинает выполняться от соответствующей директивы case и дальше, в ближайшее break (или до конца switch).
- Если ни один case не совпали - выполняется (если есть) вариант default.
- Если break нет, то выполнение пойдет ниже следующими case, при этом другие проверки игнорируются.

Группировка switch

```
let a = 2 + 2;
switch (a) {
   case 4:
     alert ('Вірно!');
     break;

   case 3: // (*)
   case 5: // (**)
     alert ('Невірно!');
     alert ('Трохи помилилися, буває.');
     break;

   default:
     alert ('Дивний результат, дуже дивний');
}
```

Практика

С помощью конструкции switch записать следующие условия:

- если ввели 1, то вывести в консоль 'a'
- если ввели 2 - "b"
- если ввели 3 - "c"
- иначе - "z"
-

Циклы

используются когда нужно несколько раз выполнить однотипные задачи
добавить числа от 1 до 5

```
var n = 0;
n+=1;
n+=2;
n+=3;
n+=4;
n+=5;

for(let i=0; i<=5; i++) {
    n+=i;
}
```

https://goo.gl/vQFbJt
https://learn.javascript.ru/while-for

```
for (let i = 0; i < 3; i++) {
  alert( i );
}
```

Цикл выполняется так:

- Начало: i = 0 выполняется один-единственный раз, во время мероприятия в цикл.
- Условие: i <3 проверяется перед каждой итерацией и при входе в цикл, если оно false, то цикл пропускается и выполняется следующий оператор после цикла.
- если условие true то выполняется alert (i).
- Шаг: i ++ выполняется после тела на каждой итерации, но перед проверкой условия.
- Далее идет шаг 2 и т.д.

Практика

- вывести в консоль с помощью цикла квадраты чисел от 1 до 9
  // 1,4,9, ... 81

Цикл while

```
while (умова) {
    // код, тело цикла
}
```

- Пример

```
let i = 0;
while (i < 3) {
  alert( i );
  i++;
}
```

https://www.w3schools.com/js/js_loop_while.asp

Цикл с после условием do..while

```
do {
  // тело цикла
} While (условие)
```

- сначала выполнится тело цикла а затем проверяется условие

```
let i = 0;
do {
  alert( i );
  i++;
} while (i < 3);
```

Остерегайтесь бесконечного цикла

- если условие всегда будет true

```
for (let i = 0; i > 0; i++) {
  alert( i );
}

while (true) {
  // ...
}
```

Практика

- вывести в консоль с помощью цикла while квадраты чисел от 1 до 9 // 1,4,9, ... 81

Прерывание цикла break

- можно выйти из цикла в любой момент

```
for (let i = 0; i < 10; i++) {
    if (i === 3) { break; }
    console.log(i);
}
```

в консоль выведется 0,1,2, выполнение цикла прекратится и перейдет к следующему кода

Следующая итерация continue

- прерывает не весь цикл, а только текущую итерацию и переходит к следующей

```
for (let i = 0; i < 10; i++) {
    if (i === 3) { continue; }
    console.log(i);
}
```

в консоль выводится 0,1,2,4,5,6,7,8,9

Функции

- применяются когда нужно выполнить код в разных частях программы

```
function showName() {          // объявление функции
    console.log('My name');
}

showName();                    // вызов функции
```

Область видимости

- переменная объявлена и инициализирована через var внутри функции имеют локальную область видимости

```
function showMessage () {
   let message = 'Hello'; // локальная переменная

   alert (message);
}
```

Доступ к внешним переменным

```
let message = ‘Pruvit’;
function showMessage () {
   message = 'Hello'; // глобальная переменная

   alert (message);
}
```

Параметры функций

- позволяет передать данные в функцию

```
function showMessage (name) {
   message = 'Hello,' + name;

   alert (message);
}
showMessage(‘Yurii’);
```

Параметры по умолчанию

```
function showMessage (name, end='!!!') {
   message = 'Hello,' + name + end;

   alert (message);
}
showMessage('Yurii'); // Yurii!!!
```

даже если не передали параметр то возьмется параметр по умолчанию

```
showMessage('Yurii', '...'); //Yurii…
```

перезаписывается параметр по умолчанию

Возвращение значения

- Функция может вернуть результат, который будет передан в момент когда она вызвана
- Для возвращения значения используется директива return
- Если в функции не указан return то она возвращает undefined
- Код return после не выполняется

```
function calc(number1, number2){
    return number1 * number2 ** 2;
}
var result = calc(4, 6);
```

Выбор именные функции

Функция - это действие. Поэтому имя функции обычно глаголом. Оно должно быть простым, точным и описывать действие функции, программист, будет читать код, получил верное представление о том, что делает функция.

Функции, начинающиеся с ...

"Get ..." - возвращают значение,
"Calc ..." - что-то вычисляют,
"Create ..." - что-то создают,
"Check ..." - что-то проверяют и возвращают логическое значение, и т.д.

Функция коллбэк

В функцию можно передать как параметр другой функции, функцию-коллбэк

```
function ask(question, yes, no) {
  if (confirm(question)) yes()
  else no();
}

function showOk() {
  alert( "Ви погоджуєтесь." );
}

function showCancel() {
  alert( "Ви відмінили." );
}


ask("Ви погоджуєтесь?", showOk, showCancel);
```

Объявления Function Expression

```
let f = function (параметры) {
   // тело функции
};
```

Объявления Function Declaration

```
function name(параметры) {
   // тело функции
};
```

Разница между двумя объявлениями

- Основное различие между ними: функции, объявленные как Function Declaration, создаются интерпретатором к выполнению кода

```
showName();
function showName() {
    console.log('My name');
}
```

Функции-стрелки

Простая и короткая запись функции

```
let sum = function(a, b) {
  return a + b;
};
// стрелочные функция
let sum = (a, b) => a + b;
```

Практика

- функция принимает два параметра (числа) и возвращает больший из них

Ссылки

https://learn.javascript.ru/ifelse if
https://learn.javascript.ru/while-for циклы
https://learn.javascript.ru/switch switch
https://learn.javascript.ru/function-basics функции
https://webref.ru/dev/learn-javascript основы js

Видео

https://www.youtube.com/watch?v=kFDhIxBV2cs циклы
https://www.youtube.com/watch?v=FZkSwa_rm_E функции