Оператори

План

- умовний оператор if
- логічні оператори
- оператор switch
- оператори циклу while, for
- функції

Оператор if

- Умовні оператори використовуються для виконання різних дій на основі різних умов. Якщо результатам обчислення умови(вираз приводиться до логічного типу) є `true` то виконується `вираз1`, якщо `false` то `вираз2`.

```
if (умова){
   вираз1}
[else {
   вираз2}]
```

У логічному контексті:

- Число 0, порожній рядок "", `null` і `undefined`, а також `NaN` є `false`,
- Решта значення - `true`.

Якщо є тільки вираз1 то можна писати без {}

```
if(true) x=1;
```

if..else

- необов’язковий блок `else` виконається якщо умова `false`

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

Кілька умов, else if

- використовується якщо потрібно перевірити кілька умов

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

Порівняння з тернарним оператором

- запис через if

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

- запис через ?

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

Практика

- написати оператор if..else який примає значення з prompt і виводить в консоль
  - 1, якщо значення більше нуля,
  - -1, якщо значення менше нуля,
  - 0, якщо значення дорівнює нулю.
- переписати в тернарний оператор
  var a = 1;

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

Логічні оператори

&&	логічне І
||	логічне АБО
!	логічне НІ

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

Логічне І (&&)

- повертає true, якщо обидва аргументи істинні, а інакше - false

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

- Якщо лівий аргумент - false, оператор && повертає його і закінчує обчислення. Інакше - обчислює і повертає правий аргумент.

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

Логічне АБО (||)

- якщо хоча б один з аргументів true, то повертає true, інакше - false "

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

- оператор АБО обчислює рівно стільки значень, скільки необхідно - до першого true.

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

Логічне НІ (!)

- Спочатку наводить аргумент до логічного типу true / false.
- Потім повертає протилежне значення.

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

- Подвійне ні використовується щоб привести значення до булевого типу

```
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

- заміняє собою кілька перевірок `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 ('Малувато');
     break;
   case 4:
     alert ('В точку!');
     break;
   case 5:
     alert ('Перебір');
     break;
   default:
     alert ('Я таких значень не знаю');
}
```

Як працює switch

- Змінна x перевіряється на строгу рівність першому значенню value1, потім другого value2 і так далі.
- Якщо відповідність встановлено - switch починає виконуватися від відповідної директиви case і далі, до найближчого break (або до кінця switch).
- Якщо жоден case не співпали - виконується (якщо є) варіант default.
- Якщо break немає, то виконання піде нижче за наступними case, при цьому інші перевірки ігноруються.

Групування 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

- За допомогою конструкції switch записати такі умови:
- якщо ввели 1, то вивести в консоль ‘a’
- якщо ввели 2 - ‘b’
- якщо ввели 3 - ‘c’
- інакше - ‘z’

Цикли

- використовуються коли потрібно декілька раз виконати однотипні задачі
- додати числа від 1 до 5

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

https://goo.gl/vQFbJt

```
n+=5;
```

https://learn.javascript.ru/while-for

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

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

Цикл виконується так:

- Початок: i = 0 виконується один-єдиний раз, під час заходу в цикл.
- Умова: i <3 перевіряється перед кожною ітерацією і при вході в цикл, якщо воно false, то цикл пропускається і виконується наступний оператор після циклу.
- якщо умова true то виконується alert (i).
- Крок: i ++ виконується після тіла на кожній ітерації, але перед перевіркою умови.
- Далі йде крок 2 і так далі

Практика

- вивести в консоль за допомогою циклу квадрати чисел від 1 до 9
  // 1,4,9,...81

Цикл while

```
while (умова) {
    // код, тіло циклу
}
```

- Приклад

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

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

Цикл з після умовою do..while

```
do {
    // тіло циклу
} while (умова);
```

- спочатку виконається тіло циклу а потім перевіриться умова

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

Остерігайтеся безкінечного циклу

- якщо умова завжди буде true

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

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

Практика

- вивести в консоль за допомогою циклу `while` квадрати чисел від 1 до 9 // 1,4,9,...81

Переривання циклу break

- можна вийти з циклу в будь-який момент

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

в консоль виведиться 0,1,2, виконання циклу припинеться і перейде до наступного коду

Наступна ітерація continue

- перериває не весь цикл, а тільки поточну ітерацію і переходить до наступної

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

в консоль виведиться 0,1,2,4,5,6,7,8,9

Функції

- застосовуються коли потрібно виконати код в різних частинах програми

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

showName();                                    // виклик функції
```

Область видимості

- змінна оголошенна і ініціалізована через `var` в середині функції мають локальну область видимості

```
function showMessage () {
   let message = 'Hello'; // локальна змінна

   alert (message);
}
```

Доступ до зовнішніх змінних

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

   alert (message);
}
```

Параметри функцій

- дозволяє передати дані в функцію

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

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

Параметри по замовчуванні

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

   alert (message);
}
showMessage('Yurii'); // Yurii!!!
```
навіть якщо не передали параметр то візьметься параметр по замовчуванні

```
showMessage('Yurii', '...'); //Yurii...
```
перезаписується параметр по замовчуванні

Повернення значення

- Функція може повернути результат, який буде переданий в момент коли вона викликана
- Для повернення значення використовується директива `return`
- Якщо у функції не вказаний `return` то вона повертає `undefined`
- Код `return` після не виконується

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

Вибір іменні функції

Функція - це дія. Тому ім'я функції зазвичай є дієсловом. Воно повинно бути простим, точним і описувати дію функції, щоб програміст, який буде читати код, отримав вірне уявлення про те, що робить функція.

Функції, що починаються з ...

"Get ..." - повертають значення,
"Calc ..." - щось обчислюють,
"Create ..." - щось створюють,
"Check ..." - щось перевіряють і повертають логічне значення, і т.д.

Функції колбеки

У функцію можна передати як параметр іншу функцію, функцію-колбеку

```
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);
```

Оголошення Function Expression

```
let f = function (параметри) {
   // тіло функції
};
```

Оголошення Function Declaration

```
function name(параметри) {
   // тіло функції
};
```

Різниця між двома оголошеннями

- Основна відмінність між ними: функції, оголошені як Function Declaration, створюються інтерпретатором до виконання коду

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

Функції-стрілки

Простіший і коротший запис функції

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

Практика

- функція приймає два параметри(числа) і повертає більший з них

Посилання

https://learn.javascript.ru/ifelse if
https://learn.javascript.ru/while-for цикли
https://learn.javascript.ru/switch switch
https://learn.javascript.ru/function-basics функції
https://webref.ru/dev/learn-javascript основи js

Відео

https://www.youtube.com/watch?v=kFDhIxBV2cs цикли
https://www.youtube.com/watch?v=FZkSwa_rm_E функції